Détection de ruptures multiples – application aux signaux physiologiques.

Charles Truong

université PARIS-SACLAY

école normale supérieure paris–saclay

NNT: 2018SACLN030

Détection de ruptures multiples – application aux signaux physiologiques

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'École Normale Supérieure

École doctorale n°574 Mathématiques Hadamard (EDMH)
Spécialité de doctorat: Mathématiques appliquées

Thèse présentée et soutenue à Cachan, le 29 novembre 2018, par

M. Charles Truong

Composition du Jury :

M. P. Gallinari Professeur, Sorbonne Université (LIP6)	Président
M. Z. Harchaoui Maître de Conférences, University of Washington	Rapporteur
M. F. Rossi Professeur, Université Paris 1 Panthéon-Sorbonne (SAMM)	Rapporteur
Mme C. Lévy-Leduc Professeure, AgroParisTech (MMIP)	Examinateur
M. N. Vayatis Professeur, École Normal Supérieure Paris-Saclay (CMLA)	Directeur de thèse
M. L. Oudre Maître de Conférences, Université Paris 13 (L2TI)	Co-Directeur de thèse

Thèse de doctorat

Contents

Acknowledgements

First, I would like to thank my advisers Nicolas Vayatis and Laurent Oudre, for giving me this unique opportunity to work and learn in such a wonderful environment. I would also like to thank Fabrice Rossi and Zaid Harchaoui for doing me the honor of reviewing this thesis. I am also grateful to Céline Lévy-Leduc and Patrick Gallinari, for participating in my thesis committee and taking interest in my work. It is an honor to have you all on my committee.

I would also like to thank all the team at CMLA for all the good moments that we shared: Julien Audiffren, Mounir Atiq, Ioannis Bargiotas, Miguel Colom, Emile Contal, Rémy Degenne, Thomas Douillet-Grellier, Mathilde Fekom, Kumar Gaurav, Pierre Humbert, Mathieu Jedor, Argyris Kalogeratos, Batiste Le Bars, Rémi Lemonnier, Zhijin Li, Myrto Limnios, Cédric Malherbe, Juan Mantilla, Steven Masfaraud, Ludovic Minvielle, Thomas Moreau, Alice Nicolaï, Vianney Perchet, Théo Saillant, Dripta Sarkar, Kevin Scaman, Asma Toumi and more generally the whole lab, Véronique Almadovar, Micheline Brunetti, Sandra Doucet, Atman Kendira, Christophe Labourdette, Delphine Laverne, Alina Müller, Virginie Pauchont, and Massil Achab. It has been a pleasure working with you over the years and I hope our paths will cross again.

I would also like to express my gratitude to all the people from the Cognac-G team with whom I had the chance to collaborate: Eric Krejci, Damien Ricard, Pierre-Paul Vidal, Alain Yelnik, Catherine De Waele as well as Rémi Barrois-Müller, Stéphane Buffat, Clément Provost, Alfredo Pulini.

Abstract

This work addresses the problem of detecting multiple change points in (univariate or multivariate) physiological signals. Well-known examples of such signals include electrocardiogram (ECG), electroencephalogram (EEG), inertial measurements (acceleration, angular velocities, etc.). The objective of this thesis is to provide change point detection algorithms that (i) can handle long signals, (ii) can be applied on a wide range of real-world scenarios, and (iii) can incorporate the knowledge of medical experts. In particular, a greater emphasis is placed on fully automatic procedures which can be used in daily clinical practice. To that end, robust detection methods as well as supervised calibration strategies are described, and a documented open-source Python package is released.

The first contribution of this thesis is a sub-optimal change point detection algorithm that can accommodate time complexity constraints while retaining most of the robustness of optimal procedures. This algorithm is sequential and alternates between the two following steps: a change point is estimated then its contribution to the signal is projected out. In the context of mean-shifts, asymptotic consistency of estimated change points is obtained. We prove that this greedy strategy can easily be extended to other types of changes, by using reproducing kernel Hilbert spaces. Thanks this novel approach, physiological signals can be handled without making assumption of the generative model of the data. Experiments on real-world signals show that those approaches are more accurate than standard sub-optimal algorithms and faster than optimal algorithms.

The second contribution of this thesis consists in two supervised algorithms for automatic calibration. Both rely on labelled examples, which in our context, consist in segmented signals. The first approach learns the smoothing parameter for the penalized detection of an unknown number of changes. The second procedure learns a non-parametric transformation of the representation space, that improves detection performance. Both supervised procedures yield finely tuned detection algorithms that are able to replicate the segmentation strategy of an expert. Results show that those supervised algorithms outperform unsupervised algorithms, especially in the case of physiological signals, where the notion of change heavily depends on the physiological phenomenon of interest.

All algorithmic contributions of this thesis can be found in `ruptures`, an open-source Python library, available online. Thoroughly documented, `ruptures` also comes with a consistent interface for all methods.

Notations

General

$\lvert X \rvert$	Number of elements of a set X
$\{y_t\}_{t=1}^{T}$	$\mathbb{R}^d$-valued signal with T samples
$\lfloor \cdot \rfloor$	Integer part function
$\mathbb{1}\{\cdot\}$	Indicator function taking values in $\{0,1\}$
$\mathbb{P}(A)$	Probability of event A
$\mathbb{E}(Y)$	Expected value of the random variable Y
$X_n \xrightarrow{p} X$	Convergence in probability of the sequence of random variables $(X_n)_n$ to the random variable X

Matrix

M'	Transpose matrix of M
$M_{i,j}$, $M_{i,\bullet}$ and $M_{\bullet,j}$	(i,j)-entry, i-th row and j-th column of a matrix M
$\lVert \cdot \rVert$	Frobenius norm
$\mathrm{tr}(\cdot)$	Trace operator
$\lVert M \rVert_{0,1}$	Number of non-zero rows of M
$M^{\dagger}$	Moore–Penrose pseudoinverse of a matrix M
$\mathbb{1}_{a \times b}$	Vector of size $a \times b$ with all entries equal to 1
I_n	Identity matrix in $\mathbb{R}^{n \times n}$
$\mathrm{diag}(M_1, M_2, \dots)$	Block diagonal matrix with blocks $M_1, M_2, \dots$

Introduction générale (en français)

1 Contexte de la thèse

Contexte général. Au cours des dernières décennies, les applications de mesure de soi sont devenues de plus en plus répandues dans la population. De nombreuses solutions commerciales sont disponibles pour calculer des quantités liées à la santé telles que le nombre de pas, la distance parcourue, les variations de poids, l'énergie dépensée, etc. Ce changement remarquable des habitudes de santé est rendu possible par la progression stupéfiante des capteurs intégrés aux appareils mobiles. Dans le contexte médical, la même tendance a été observée, grâce à la montée en puissance de capteurs bon marché, faciles à porter et à manipuler. En conséquence, de plus en plus de signaux physiologiques, tels que l'électrocardiogramme (ECG), l'électroencéphalogramme (EEG), les accélérations du corps et des membres, peuvent être collectés. Cette tendance présente de nombreux avantages: faciliter le diagnostic précoce, promouvoir la télémédecine, rendre les soins de santé plus abordables, etc. Avant de récolter les fruits de cette avancée technologique, il convient de concevoir des méthodes automatiques et objectives pour extraire des informations de cet important volume de données brutes. La transition de signaux bruts à des données intelligibles est cruciale pour le succès de la médecine nouvelle génération et est devenue un véritable sujet d'intérêt.

Collaboration avec Cognac-G. Au cours de ma thèse, j'ai collaboré avec Cognac-G, une équipe de recherche regroupant des chercheurs en apprentissage statistique et des chercheurs en médecine, réunis autour de la quantification du comportement humain et animal. À cette fin, plusieurs protocoles expérimentaux ont été développés pour un large éventail de problèmes cliniques allant de la respiration de souris ou de la locomotion humaine aux mouvements oculaires du nourrisson. Chaque protocole est surveillé avec un ou plusieurs capteurs afin de fournir une quantification objective du phénomène d'intérêt. Les séries temporelles résultantes (univariées ou multivariées), appelées signaux physiologiques, sont ensuite étudiées. Le premier défi consiste à extraire des informations pertinentes de ces signaux, afin de les interpréter et d'aider à comprendre les mécanismes physiologiques, biologiques ou biomécaniques qui les ont produits. Le deuxième défi consiste à automatiser le processus de quantification afin de fournir des outils pouvant être utilisés par les médecins pour le suivi longitudinal et la comparaison interindividuelle de leurs patients.

2 Motivations

2.1 Comprendre les données brutes

Dans la grande majorité des situations, le contexte clinique impose des contraintes pratiques au processus d'acquisition des données. Par exemple, les cliniciens n'ont peut-être pas la possibilité d'activer et de désactiver librement les capteurs, au début et à la fin du phénomène considéré. En outre, les sujets surveillés peuvent être invités à effectuer successivement différentes activités physiques. Dans ces situations, les signaux collectés sont constitués de phases consécutives et les informations précises sur le début et la fin de chaque phase ne sont pas toujours disponibles. La *segmentation de signal* ou *détection de ruptures* est une étape cruciale pour pré-traiter une grande quantité de séries temporelles. (Les deux termes sont utilisés de manière équivalente.) Elle consiste à trouver les limites temporelles des régimes successifs du signal, pour pouvoir les supprimer ou les analyser. Cette étape est essentielle pour la contextualisation de longues séries temporelles.
Dans la pratique clinique quotidienne, cette situation se rencontre souvent, lorsque des sujets sont surveillés pendant qu'ils suivent un protocole médical. Généralement, un clinicien demande à un patient d'effectuer plusieurs exercices physiques consécutifs pendant que certains capteurs enregistrent certaines variables physiologiques et biomécaniques (par exemple, la fréquence cardiaque, l'absorption d'oxygène, l'accélération du corps). Afin de quantifier l'évolution du patient au cours du protocole, le signal de surveillance est segmenté, ce qui signifie qu'il est divisé en sous-signaux, chacun correspondant à une phase cohérente (par exemple, un seul exercice). Certaines caractéristiques intéressantes sont ensuite calculées pour chaque phase. Ce schéma est utilisé pour l'étude de la locomotion humaine (plus précisément, l'analyse de la marche) présentée plus loin dans ce manuscrit et illustrée sur la figure 0.1.
D'un point de vue pratique, la segmentation du signal peut être réalisée manuellement par les cliniciens. Par exemple, ils peuvent enregistrer les temps de début et de fin des phénomènes d'intérêt, ou marquer le moment des changements en analysant la série temporelle brute. Cependant, les deux approches peuvent s'avérer fastidieuses si le protocole est complexe et nécessite toute l'attention du clinicien, ou si les modifications ne sont pas facilement visibles à partir des signaux bruts. En effet, même si des spécialistes qualifiés sont en mesure d'évaluer l'état du patient à l'œil nu, il est difficile d'appliquer cette expertise sur des séries temporelles. De même, lors des essais cliniques, il peut être essentiel de réduire le nombre d'opérations manuelles, qui sont sujettes à interprétation et donc subjectives.
Ceci motive l'étude des méthodes de segmentation automatique du signal. Pour faire face à la diversité des séries temporelles physiologiques, des algorithmes robustes et polyvalents sont nécessaires, ainsi que des procédures systématiques pour les calibrer. L'objectif est de capturer l'expertise médicale des cliniciens, avec le moins d'intervention humaine possible. De plus, les algorithmes ne doivent générer que peu ou pas de surcoût informatique s'ils doivent être utilisés en routine clinique quotidienne.

2.2 Exemples motivants

Nous décrivons maintenant trois exemples motivants, tous issus d'une coopération entre des chercheurs en apprentissage statistique et des chercheurs en médecine de Cognac-G. Un accent particulier est mis sur le premier exemple (analyse de la marche humaine) car il

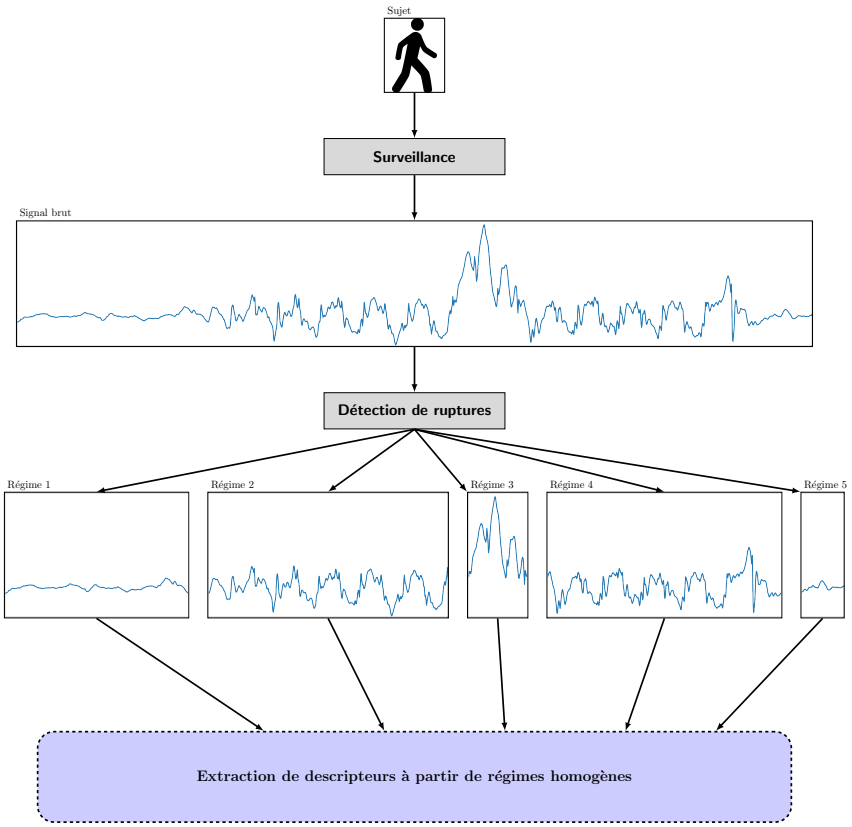


Figure 0.1: Schéma d'une étude médicale pour l'analyse de la marche.

est présent tout au long de ce manuscrit et constitue la pierre de touche de la précision de la segmentation pour les contributions de cette thèse.

2.2.1 Analyse de la marche humaine

Contexte. Le mouvement humain résulte d'un processus complexe, qui nécessite la coordination de nombreux muscles. Certaines pathologies (telles que la maladie de Parkinson, l'arthrite, les accidents vasculaires cérébraux, l'obésité, le diabète,...) peuvent altérer la locomotion, augmenter le risque de chute et menacer l'autonomie des patients. La quantification et l'évaluation objectives de la locomotion est donc un problème crucial qui a été abordé dans la littérature en mesurant le mouvement à l'aide de plusieurs types de capteurs tels que des capteurs inertiels, des tapis instrumentés, des plates-formes de force, un système de suivi caméra-optique ou des résistances sensibles à la force placées à l'intérieur de semelles. Les signaux obtenus à partir de ces capteurs sont traités (automatiquement ou manuellement) afin d'extraire certaines caractéristiques qui caractérisent la locomotion (vitesse, variabilité, régularité, ...).

Protocole. Dans ce contexte, un protocole clinique a été conçu et mis en œuvre au sein de Cognac-G pour l'étude de la locomotion humaine à l'aide d'unités de mesure inertielle, composées d'accéléromètres 3D, de gyroscopes 3D et de magnétomètres 3D. Ils sont relativement peu coûteux, ne nécessitent pas de salle réservée aux expériences et que

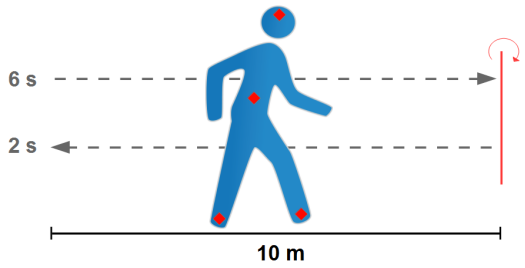
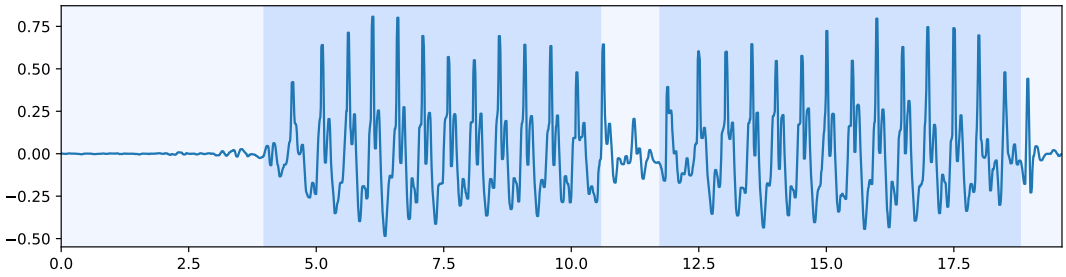
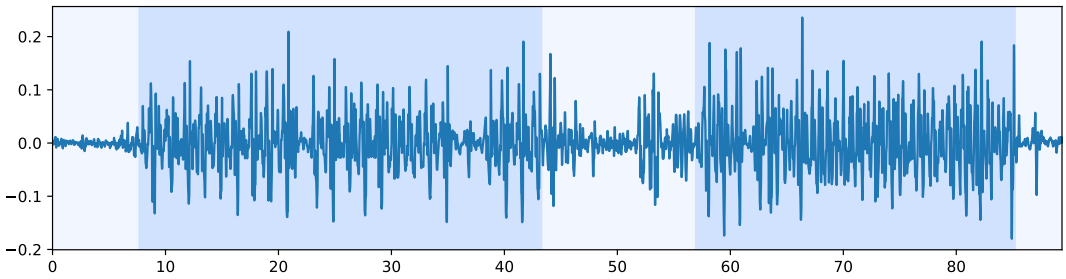


Figure 0.2: Schéma du protocole utilisé pour l'analyse de la marche humaine. Les points rouges indiquent les positions des capteurs.



(a) Sujet sain.



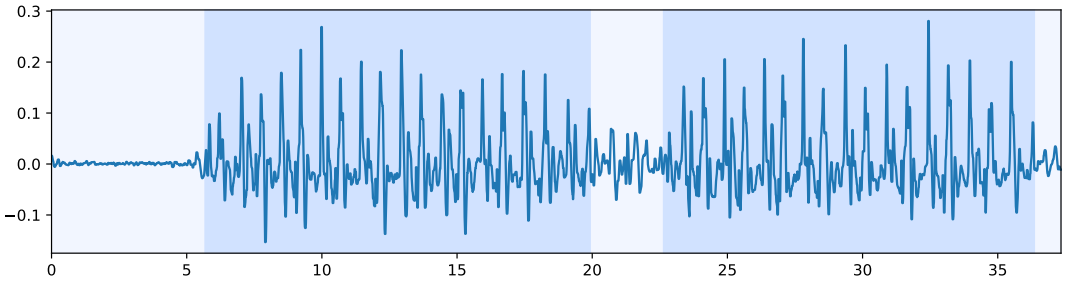
(b) Patient atteint d'arthrose.

Figure 0.3: Accélération verticale (m/s^2) en fonction du temps (s), pour le capteur placé au bas du dos, pour deux sujets différents. Les couleurs alternées marquent les phases consécutives: «Debout», «Marche», «Demi-tour», «Marche» et «Arrêt».

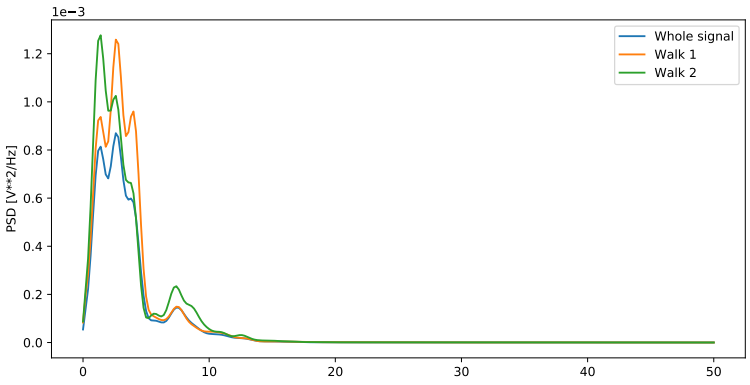
leur petite taille les rend faciles à manipuler dans des situations cliniques quotidiennes.Les données utilisées pour la conception et l'évaluation des méthodes présentées dans cette thèse ont été fournies par les services médicaux suivants: Service de chirurgie orthopédique et de traumatologie de l'Hôpital Européen Georges Pompidou, Assistance Publique des Hôpitaux de Paris, Service de médecine physique et de réadaptation de l'Hôpital Fernand Widal, Assistance Publique des Hôpitaux de Paris, Service de neurologie de l'Hôpital d'Instruction des Armées du Val de Grâce, Service de Santé des Armées. L'étude a été validée par un comité d'éthique local et tous les sujets ont donné leur consentement écrit pour y participer. Tous les signaux ont été acquis à 100 Hz avec des capteurs XSens^{TM} sans fil situés dans le bas du dos et fixés à l'aide d'une bande velcro conçue par XSens^{TM}. On a demandé à tous les sujets de rester immobiles pendant 6 secondes, de marcher 10 mètres à la vitesse de marche préférée sur une surface plane, de faire demi-tour, de revenir en arrière et de rester immobiles pendant 2 secondes (voir Figure 0.2). Deux exemples de

signaux enregistrés au cours de ce protocole sont visibles sur la figure 0.3. Les deux phases plates aux extrémités du signal correspondent aux périodes pendant lesquelles le sujet est immobile. Les motifs répétés représentent les pas. En fonction de la pathologie (ou son absence), la longueur du signal varie de 20 secondes à 90 secondes. En outre, les pas et le demi-tour sont moins visibles sur le signal pour le patient atteint d'arthrose.

Application de la détection de rupture. Les propriétés spectrales des signaux collectés peuvent fournir aux cliniciens des informations adaptées à l'analyse de la marche [26]. En effet, comme indiqué sur la figure 0.4, les phases de marche montrent une structure harmonique forte. Cependant, en comparant la densité spectrale de puissance (psd) de chaque régime «Marche», des différences avec la psd de l'ensemble du signal sont observées. Cela est dû au fait que la structure harmonique de l'enregistrement complet est corrompue par des phases non périodiques, à savoir «Debout», «Demi-tour» et «Arrêt». De plus, les deux régimes «Marche» (aller et retour) n'ont pas la même distribution de fréquence. Plus précisément, les pics de fréquence pour les premier et deuxième «Marche» sont respectivement situés autour de 2,6 Hz et de 1,4 Hz. Ce décalage de fréquence, qui peut servir à mesurer la fatigue du sujet, ne peut être détecté que grâce à la segmentation du signal.



(a) Accélération verticale (m/s^2) en fonction du temps (s).



(b) Densité spectrale de puissance en fonction de la fréquence (Hz).

Figure 0.4: Exemple de signal. (Haut) Accélération verticale (m/s^2) du capteur (placé au bas du dos). Les couleurs alternées marquent les phases consécutives: «Debout», «Marche», «Demi-tour», «Marche» et «Arrêt». (Bas) Densité spectrale de puissance pour le signal complet, la première phase «Marche» et la seconde phase «Marche».

2.2.2 Autres exemples de signaux physiologiques

Deux autres exemples de collaborations au sein de Cognac-G sont présentés. Dans les deux cas, la segmentation du signal est une étape de prétraitement cruciale.

Analyse du contrôle respiratoire chez la souris. L'acétylcholine (ACh) est un neurotransmetteur (un produit chimique organique libéré par le système nerveux pour envoyer des signaux) activant les muscles qui participent à un grand nombre de fonctions corporelles, parmi lesquelles la respiration. Les substances cholinergiques (qui altèrent la capacité de libération ou d'activation de l'ACh) peuvent être retrouvées dans de nombreux médicaments, toxines et agents nerveux chimiques, et peuvent entraîner des défaillances respiratoires. La quantification et l'évaluation objectives des comportements respiratoires lors d'une crise cholinergique peuvent permettre de mieux comprendre l'influence de l'ACh sur le contrôle respiratoire. Dans la littérature, cette question a été abordée en surveillant les souris après leur exposition à des mélanges de gaz spécifiques.
Dans ce contexte, un protocole a été conçu et mis en œuvre par des chercheurs de Cognac-G, conformément aux lois de protection des animaux de l'Union européenne et du gouvernement français. Plusieurs variables physiologiques liées à la respiration (telles que la durée d'inspiration et d'expiration, la fréquence respiratoire,…) sont enregistrées chez des souris présentant des déficits particuliers des fonctions cholinergiques. Les souris ont été placées dans une chambre de pléthysmographie (une chambre scellée utilisée pour mesurer les changements de volume dans les poumons) pendant 15 à 20 minutes, puis sorties pour être exposées au gaz et replacées dans la chambre. Un exemple illustratif est présenté sur la figure 0.5. La première période du signal (les 20 premières minutes) fournit les valeurs de référence pour les variables enregistrées. Dans la seconde période du signal, différents régimes peuvent être observés, correspondant à différents états de la souris: calme au début, puis stressée avec des difficultés respiratoires prononcées.

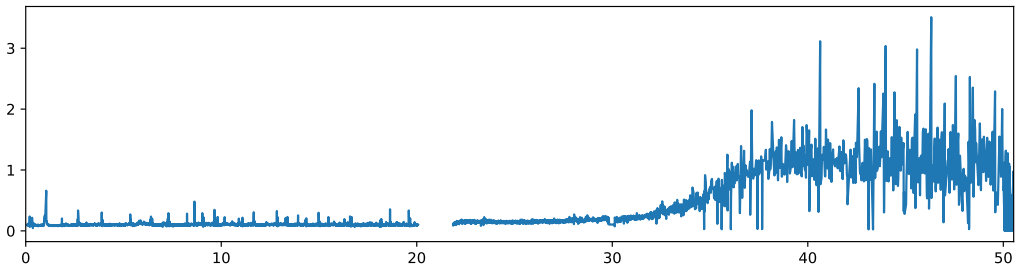


Figure 0.5: Évolution de la durée d'expiration (en seconde) d'une souris, en fonction du temps, en minute (fréquence d'échantillonnage 5 Hz). Après 20 minutes, la souris est exposée à un mélange de gaz spécifique. La période sans échantillon correspond à la manipulation de la souris.

Analyse de conscience pendant l'anesthésie. L'anesthésie générale consiste en un coma médicalement provoqué et est obligatoire pour certaines chirurgies. Pour réduire les risques associés à une telle procédure, il est courant de surveiller la profondeur de conscience des patients au moyen d'électroencéphalogrammes (EEG). L'état du patient (éveillé, sous sédation,…) est ensuite déduit à l'aide d'un algorithme approprié. L'objectif

est de prévenir tout événement indésirable pendant et après la chirurgie et de réduire la consommation de médicaments. Cependant, l'analyse des signaux enregistrés souffre de plusieurs limitations, parmi lesquelles la sensibilité de l'EEG aux appareils électroniques externes.
Dans ce contexte, Cognac-G développe un protocole de suivi des patients anesthésiés. Il consiste en plusieurs dispositifs (notamment un EEG, un électrocardiogramme, un oxymètre de pouls) et des traitements algorithmiques des signaux collectés. L'un des problèmes que ce protocole vise à résoudre est la détection d'artefacts de champ électromagnétique externe dans l'environnement. Un exemple illustratif est présenté sur la figure 0.6. Plusieurs échantillons aberrants et périodes de forte amplitude de tension peuvent être observés dans cet enregistrement. Ils sont la conséquence des dispositifs électro-chirurgicaux utilisés pour couper et cautériser les tissus. La suppression de ces périodes est une étape cruciale pour déduire l'état du patient.

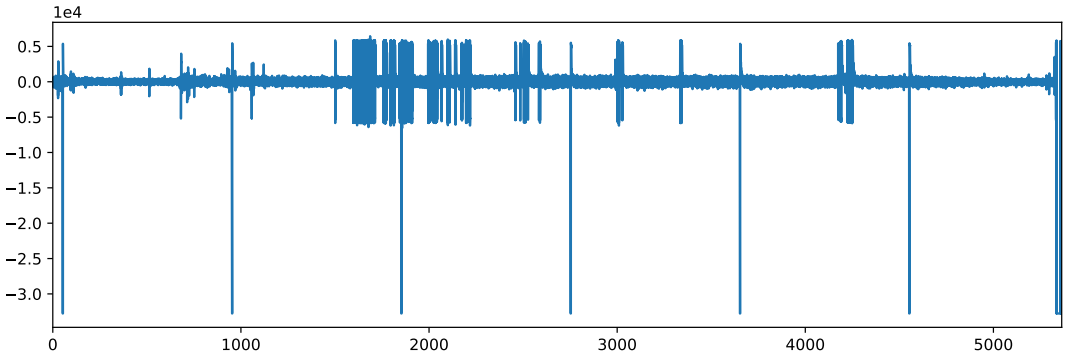


Figure 0.6: Électroencéphalogrammes (μV) en fonction du temps (s) d'un patient pendant une anesthésie (fréquence d'échantillonnage 100 Hz).

3 Détection de ruptures pour les données physiologiques

Plusieurs principes généraux ont émergé du contexte décrit précédemment. Ces principes ont fortement influencé la forme des algorithmes de détection de points de changement proposés dans ce manuscrit. Ils nous ont notamment amenés à envisager des stratégies basées sur l'apprentissage supervisé.

3.1 Principes généraux

Nous énumérons les principes généraux qui servent de lignes directrices pour les contributions algorithmiques de cette thèse.

Coût de calcul et robustesse. La surveillance d'un sujet soumis à un protocole spécifique peut générer un volume considérable de données brutes. En effet, un protocole peut durer longtemps (dans Cognac-G, ils vont de quelques secondes à quelques heures), les capteurs peuvent avoir une fréquence d'échantillonnage élevée pour capturer un phénomène de haute fréquence (jusqu'à 1000 Hz dans les études ophtalmologiques) ou il peut y avoir de nombreux capteurs collectant simultanément plusieurs caractéristiques physiologiques.

Cependant, dans la pratique clinique courante, les cliniciens s'attendent à ce que les informations extraites des données brutes soient rapidement disponibles, de sorte qu'elles puissent être visualisées et interprétées dans les délais de consultation. De plus, pour des raisons pratiques et juridiques, les ressources de calcul nécessaires à l'exécution des algorithmes sont souvent limitées aux ordinateurs portables ou aux périphériques intégrés des cliniciens. En conséquence, pour traiter les données brutes dans un délai raisonnable, nous devons envisager des méthodes de segmentation sous-optimales, aux temps de calcul restreints. En contrepartie, ces algorithmes sont moins robustes que les méthodes optimales. Par conséquent, dans cette thèse, nous nous concentrons sur le développement d'un algorithme sous-optimal, capable de gérer les contraintes en temps de calcul tout en conservant la majeure partie de la robustesse des procédures optimales.

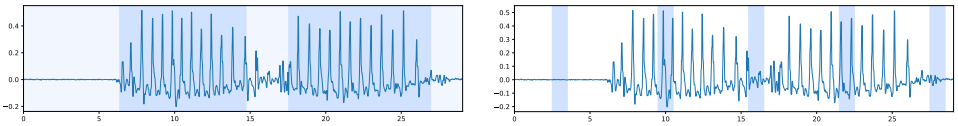
Versatilité. Au sein de l'écosystème Cognac-G, le type de signaux collectés varie considérablement. Cela est dû au fait qu'un grand nombre de sujets ont été surveillés et que plusieurs paramètres de surveillance sont utilisés, ce qui donne un large éventail de valeurs de paramètres cliniques (âge, poids ou pathologie, par exemple) et de capteurs. Selon le contexte, les signaux peuvent avoir des caractéristiques (dimensions, des taux d'échantillonnage, des unités, des distributions, etc.) très différentes. En raison de cette variabilité, nous ne pouvons considérer qu'une classe générale de méthodes de segmentation qui en font que peu, voire aucune hypothèse, sur la forme du modèle de signal sous-jacent ou sur l'emplacement des points de rupture. De plus, ces algorithmes doivent pouvoir détecter un nombre de changements connu ou inconnu, car les deux situations peuvent se produire dans notre contexte.

Calibration automatique. Habituellement, les collaborations au sein de Cognac-G commencent par une longue période de discussion entre chercheurs en mathématiques appliquées et chercheurs en médecine. Au cours de cette période, un modèle statistique approprié est choisi et calibré en fonction des données disponibles. Ce processus peut être difficile et prendre beaucoup de temps, en fonction de la formation statistique des cliniciens et de la formation médicale des statisticiens. En effet, même si les cliniciens comprennent parfaitement le phénomène physiologique surveillé, la traduction de leur expertise médicale en termes statistiques est un exercice complexe. L'un des objectifs de ce travail est de progresser vers une automatisation de cette opération. Plus précisément, nous visons à concevoir des mécanismes pour calibrer automatiquement les méthodes de segmentation, éliminant ainsi la nécessité d'un réglage manuel fastidieux des paramètres (souvent effectué par tâtonnement).

3.2 Apprendre d'experts

En conséquence des principes généraux décrits ci-dessus, nous proposons dans cette thèse d'appliquer des procédures d'apprentissage supervisé à la détection de ruptures. De manière générale, de telles procédures peuvent déduire des règles de décision complexes uniquement à l'aide d'exemples pertinents. Dans notre contexte, les cliniciens sont en mesure de fournir de tels exemples: ils consistent en des signaux, segmentés manuellement par les cliniciens. Notre objectif est d'utiliser ces exemples pour concevoir une méthode de détection capable de reproduire la stratégie de segmentation des cliniciens. Ici, deux types d'annotations (ou étiquettes, dans la terminologie de l'apprentissage supervisé) sont considérés: complet

et partiel. Pour un *signal complètement annoté*, les temps de toutes les ruptures sont fournis par un expert. Pour un *signal partiellement annoté*, l'expert marque uniquement certaines parties du signal comme homogènes, c'est-à-dire qu'elles ne contiennent aucun point de changement. Dans cette situation, les emplacements exacts des modifications ne sont pas connus. Les deux types d'annotations sont illustrés sur la figure 0.7. Deux scénarios d'utilisation sont couramment rencontrés dans le contexte de Cognac-G. Dans le premier scénario, un clinicien fournit un ou plusieurs signaux annotés (complètement ou partiellement). L'objectif est de segmenter tout nouveau signal issue du même contexte clinique. À cette fin, un algorithme de détection est calibré, de manière supervisée, à l'aide de l'ensemble (d'apprentissage) de signaux d'annotation, puis appliqué à de nouveaux signaux. Dans le second scénario, un clinicien fournit un ou plusieurs signaux partiellement annotés. L'objectif est de compléter les segmentations partielles de ces signaux, c'est-à-dire de récupérer tous les points de rupture. Ici, un algorithme de détection est calibré, de manière supervisée, sur chaque signal (partiellement annoté), puis appliqué sur le même signal.



(a) Annotation complète (b) Annotation partielle

Figure 0.7: Deux types d'annotation (pour un signal utilisé pour l'analyse de la marche). (Gauche) Les couleurs alternées marquent les régimes consécutifs. (Droite) Les zones bleues désignent des parties de signaux considérées comme homogènes (ne contenant aucun changement).

4 Cadre mathématique pour la détection de rupture

Dans cette thèse, nous nous concentrons sur la tâche de détection de rupture. Nous proposons une formulation générale dans laquelle nous considérons un processus aléatoire multivarié non stationnaire $y = \{y_1, \dots, y_T\}$ à valeurs dans $\mathbb{R}^d$ ($d \geq 1$). Le signal y est supposé stationnaire par morceaux, ce qui signifie que certaines caractéristiques du processus changent abruptement à des instants inconnus $t_1^\star < t_2^\star < \cdots < t_K^\star$. La détection de rupture consiste à estimer les indices $t_k^\star$. Selon le contexte, le nombre K de ruptures peut être connu ou inconnu, auquel cas il doit également être estimé. Il est important de noter que, dans notre contexte, les deux situations sont d'importance égale. En effet, le nombre de changements est souvent déterminé par le protocole de récolte des données. De plus, comme dans les applications qui ont motivé cette thèse, l'analyse du signal est toujours effectuée *a posteriori*, nous nous concentrons donc sur la détection dite hors ligne (également rétrospective ou *a posteriori*). Inversement, le cadre dit en ligne, dans lequel les échantillons du signal sont révélés progressivement, a été introduit pour l'analyse de signal en temps réel et dépasse notre cadre.

La littérature sur la détection de ruptures est riche et les applications, nombreuses. Les premiers travaux sur ce sujet remontent aux années 50 [130, 131]: le but était de localiser un

saut dans la moyenne des variables gaussiennes indépendantes et identiquement distribuées (i.i.d.). La principale application de ces méthodes était le contrôle de qualité industrielle. Depuis lors, ce problème a été activement étudié, donnant lieu à une littérature riche et diversifiée. Nous donnons des références à des recueils de synthèse complets consacrés exclusivement à la détection de ruptures et couvrant une grande partie de ce domaine de recherche, d'un point de vue paramétrique et non paramétrique [29, 36, 45, 52, 102]. (Plus de références sont données dans le chapitre 2.)

4.1 Méthodes de détection génériques

Une partie de nos contributions couvre le choix et la calibration des méthodes de détection de ruptures. L'objectif est de rendre cette étape cruciale automatique pour un praticien qui n'est pas un expert en modélisation de séries temporelles non stationnaires. À cette fin, nous formulons les hypothèses méthodologiques nécessaires sur la forme des méthodes de détection. Ceci définit un cadre qui englobe de nombreuses procédures de la littérature et a été explicitement utilisé dans plusieurs contributions pratiques [98, 121, 139] ainsi que théoriques [107, 110].

Notations. Dans la suite du chapitre, nous utilisons les notations suivantes. Pour un signal donné $y = \{y_t\}_{t=1}^{T}$, le sous-signal $\{y_t\}_{t=a+1}^{b}$ $(1 \leq a < b \leq T)$ est simplement noté $y_{a..b}$; le signal complet est donc $y = y_{0..T}$. Un ensemble d'indices est désigné par une lettre calligraphique: $\mathcal{T} = \{t_1, t_2, \dots\} \subset \{1, \dots, T\}$ et son cardinal est $|\mathcal{T}|$. Pour un ensemble d'indices $\mathcal{T} = \{t_1, \dots, t_K\}$, les indices factices $t_0 := 0$ et $t_{K+1} := T$ seront implicitement définis.

Stratégie méthodologique. La détection de ruptures est un outil d'analyse de séries temporelles non stationnaires, à l'aide de modèles simples sur chaque sous-segment. Le but est de trouver des périodes d'homogénéité dans le comportement du signal, ou de manière équivalente, pour trouver des moments de changement. Lorsqu'elle est exprimée comme un problème de sélection de modèle, la détection de ruptures revient à choisir la meilleure segmentation possible $\mathcal{T}$ selon un critère quantitatif $V(\mathcal{T}, y)$ à minimiser. (La fonction $V(\mathcal{T}, y)$ est simplement notée $V(\mathcal{T})$ quand il est évident d'après le contexte qu'elle fait référence au signal y.) Le choix d'un critère $V(\cdot)$ dépend de connaissances préalables de la tâche à accomplir. Dans ce travail, nous supposons que le critère $V(\mathcal{T})$ pour une segmentation particulière est la somme des coûts de tous les segments qui définissent la segmentation :

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

où $c(\cdot)$ est une fonction de coût qui mesure la qualité d'ajustement sur le sous-signal $y_{t_k..t_{k+1}} = \{y_t\}_{t_k+1}^{t_{k+1}}$ pour modèle donné. La "meilleure segmentation" $\widehat{\mathcal{T}}$ est celle qui minimise le critère $V(\mathcal{T})$. En pratique, les méthodes de détection se répartissent en deux catégories selon que le nombre de ruptures K est connu ou non. Si K est connu (grâce à une connaissance *a priori*), l'estimation de l'indice $\widehat{\mathcal{T}}$ est le minimum du problème d'optimisation discrète.

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}). \tag{2}$$

Si K est inconnu et doit être estimé, l'estimation de la rupture $\widehat{\mathcal{T}}$ est le minimum du problème d'optimisation discrète pénalisé:

$$\min_{\mathcal{T}} \; V(\mathcal{T}) \; + \; \text{pen}(\mathcal{T}) \tag{3}$$

où $\text{pen}(\mathcal{T})$ est une mesure appropriée de la complexité d'une segmentation $\mathcal{T}$. Toutes les méthodes de détection de ruptures considérées dans ce travail donnent une solution exacte ou approximative à (2) ou à (3), avec la fonction $V(\mathcal{T}, y)$ qui respecte le format (1).

Exemples. Parmi les exemples de procédures qui suivent ce format, on trouve l'estimation par maximum de vraisemblance (EMV) et la régression par morceaux. Dans le contexte de EMV, on suppose que les échantillons suivent une distribution $P(\cdot|\theta)$ paramétrée par θ. Entre deux ruptures, le paramètre θ est constant et il n'y a pas de dépendance entre les segments. Ici, nous aimerions détecter le moment où la valeur du paramètre θ change. La EMV est exécuté en résolvant soit les formulations (2) ou (3), en définissant la fonction de coût $c(\cdot)$ égale à la fonction de probabilité de journal négative minimale, ce qui signifie cette

$$c(y_{a..b}) := \min_{\theta} \left[-\log P(y_{a..b}|\theta) \right]. \tag{4}$$

En régression (paramétrique ou non paramétrique), le signal y est appelé une variable de réponse et nous considérons un signal $x = \{x_t\}_{t=1}^{T}$ de variables explicatives : l'objectif est de rechercher une fonction (le predicteur) f dans un sous-ensemble d'un espace fonctionnel (par exemple, des fonctions linéaires, des fonctions polynomiales, etc.) telle que les résidus carrés $\|y_t - f(x_t)\|^2$ soient aussi aussi petit que possible. En régression par morceaux, le prédicteur peut changer d'un segment à l'autre. Pour rechercher les ruptures associées, plusieurs régression sont effectuées séparément sur chaque segment. En définissant la fonction de coût $c(\cdot)$ comme égale à la somme des résidus au carré

$$c(y_{a..b}) := \min_{f} \left[\sum_{t=a+1}^{b} \|y_t - f(x_t)\|^2 \right], \tag{5}$$

les problèmes de détection de ruptures (2) et (3) sont équivalents à la régression par morceaux.

Limitations. Le cadre décrit, aussi général soit-il, est introduit pour traiter les problèmes rencontrés dans Cognac-G. Certaines approches ne rentrent pas dans ce cadre. En particulier, les approches bayésiennes ne sont pas prises en compte dans la suite de cette thèse, même si elles fournissent des résultats de pointe dans plusieurs domaines, tels que le traitement de la parole et du son. L'algorithme bayésien le plus connu est le modèle de Markov caché (HMM) [138]. Ce modèle a ensuite été étendu, par exemple avec les processus de Dirichlet [99, 123] ou les modèles produit de partition [27, 28]. Une caractéristique commune des méthodes bayésiennes est la nécessité de spécifier un *a priori* sur l'emplacement des points de rupture. Cependant, dans le contexte de Cognac-G, ces *a priori* ne se sont pas justifiés expérimentalement. Le lecteur intéressé pourra trouver des revues d'approches bayésiennes dans [36] et [45].

4.2 Conception de méthodes de détection

Les hypothèses précédentes ont une conséquence importante sur la structure des méthodes. Les méthodes de détection de point de changement qui respectent le format (2) ou (3) sont caractérisées par trois éléments.

Type de ruptures. Le type de rupture qu'une méthode est capable de détecter est caractérisé par sa *fonction de coût* $c(\cdot)$. Par exemple, définir la fonction de coût comme dans (4) permet à une méthode de détecter les changements de la valeur du paramètre θ de la distribution $P(\cdot|\theta)$. La recherche de la fonction de coût pertinente pour une tâche donnée est une partie essentielle de l'analyse qu'un expert en détection de ruptures doit effectuer.

Nombre de ruptures. Le nombre de ruptures à détecter peut être connu (grâce à des informations *a priori*) ou inconnu. Lorsqu'il est inconnu, il doit être estimé avec les emplacements des ruptures. À cette fin, un *pénalité de complexité* $\text{pen}(\cdot)$ (3) est introduit pour équilibrer le terme d'ajustement $V(\mathcal{T}, y)$. La complexité peut par exemple être proportionnelle au nombre de ruptures. Dans ce cas, la pénalité est appelée «linéaire». Le choix de la pénalité de complexité est lié à l'amplitude des changements à détecter: avec une pénalité trop «petite» (comparée à la qualité d'ajustement) dans (3), beaucoup de changements sont détectés, même ceux qui résultent d'un bruit. À l'inverse, trop de pénalisation ne permet la détection que les changements les plus significatifs, voire aucun.

Temps de calcul La dernière brique d'une procédure de détection est la *méthode de recherche* : c'est la procédure de résolution des problèmes d'optimisation (2) et (3). Dans (2), la minimisation est effectuée sur l'ensemble $\{\mathcal{T} \text{ s.c. } |\mathcal{T}| = K\}$ qui contient $\binom{T-1}{K-1}$ éléments. Dans (3), la minimisation est effectuée sur l'ensemble $\{\mathcal{T} \text{ s.c. } 1 \leq |\mathcal{T}| < T\}$ qui contient $\sum_{K=1}^{T-1} \binom{T-1}{K-1}$ éléments. Dans les deux cas, une énumération exhaustive de toutes les segmentations possibles n'est pas prohibitive en pratique. La littérature contient plusieurs méthodes pour résoudre efficacement ces problèmes, de manière exacte [30, 98] ou approximativement [29, 146]. Chaque méthode réalise un certain équilibre entre complexité de calcul et précision. Dans cette thèse, nous nous concentrons sur la fourniture d'un algorithme rapide sous-optimal capable de conserver la majeure partie de la robustesse des procédures optimales.

5 Contributions

Nous résumons les contributions de cette thèse comme suit.

- **Détection gloutonne de ruptures.** Nous développons une nouvelle méthode de recherche gloutonne pour la détection de rupture qui représente un compromis entre précision et rapidité d'exécution. Il s'agit d'une approximation séquentielle de méthodes de détection exactes (pour un nombre de modifications connu et inconnu) pouvant être combinées à des fonctions de coût basées sur un noyau. Nous prouvons sa consistance asymptotique et montrons qu'il est plus précis que les méthodes approximatives standard et plus rapide que les méthodes exactes, sur des signaux simulés et réels.

- **Détection supervisée de ruptures.** L'objectif est de fournir des procédures automatiques pour choisir et calibrer des méthodes de détection pour une tâche donnée. À cette fin, nous développons des méthodes supervisées qui reposent sur un ensemble de signaux annotés d'apprentissage, ce qui signifie que les emplacements des ruptures ont été localisés manuellement au préalable.

  - **Calibrer le paramètre de lissage par apprentissage supervisé.** Cette contribution couvre l'un des éléments déterminants des méthodes de détection: la pénalité de complexité. Un algorithme est développé pour calibrer la valeur du paramètre de lissage pour les fonctions de pénalité linéaires, de manière supervisée. Nous utilisons une formulation convexe qui reste valable pour toutes les fonctions de coût, par opposition aux autres méthodes supervisées de la littérature.
  - **Apprentissage de métrique.** Cette contribution traite d'un autre composant des méthodes de détection: la fonction de coût. Une fonction de coût, paramétrée par une pseudo-métrique de type Mahalanobis, est apprise à partir d'un ensemble de signaux annotés. La procédure d'apprentissage de métrique choisie peut également être combinée à un noyau, afin de fournir un traitement non linéaire des échantillons de signaux. Une fois que la fonction de coût est apprise, elle peut être utilisée avec n'importe laquelle des méthodes de recherche et des pénalités de complexité de la littérature.

- **Revue de littérature via une implémentation Python.** Nous développons une bibliothèque Python, appelée `ruptures`, dédiée à la détection de ruptures. Elle s'insère dans au cadre introduit précédemment et s'appuie fortement sur la revue de littérature de cette thèse. Les fonctions de coût, les méthodes de recherche et les contraintes de complexité sont codées séparément, de sorte que tous ces éléments peuvent être combinés et utilisés pour créer une méthode de détection. Grâce à son interface modulaire et cohérente, de nombreux algorithmes peuvent être exécutés simplement en modifiant quelques paramètres.
  Une documentation complète est disponible sur ctruong.perso.math.cnrs.fr/ruptures.
  Le code peut être trouvé en ligne à l'adresse reine.cmla.ens-cachan.fr[1].

6 Vue d'ensemble du manuscrit

Le reste du manuscrit est structuré en quatre parties.

- Part I : Literature review and evaluation framework
  - Chap. 2 : A selective review of change point detection methods. Il s'agit d'une revue de littérature dans laquelle sont décrits des algorithmes de pointe, d'un point de vue algorithmique et théorique. Les lecteurs peuvent trouver des tableaux de synthèse: tableau 2.1 on page 63 et tableau 2.2 on page 76
  - Chap. 3 : Evaluation framework: metrics and data sets. Ce chapitre présente le cadre d'évaluation dans lequel les performances des méthodes de détection

[1] https://reine.cmla.ens-cachan.fr/c.truong/ruptures/repository/latest/archive.zip

sont évaluées. Les ensembles de données et les mesures décrits dans ce chapitre sont utilisés tout au long de ce manuscrit.

De nombreuses notations et paramètres expérimentaux sont définis dans ces deux chapitres et utilisés dans le reste du manuscrit.

- Part II : Greedy change point detection.
  - Chap. 4 : Greedy change point detection. Une méthode gloutonne de détection de ruptures est décrite. Son objectif est de trouver un compromis entre précision et rapidité d'exécution pour la segmentation des signaux de marche.
  - Chap. 5 : Greedy kernel change point detection. Une extension, basée sur des noyaux, de l'algorithme précédent est présentée. L'objectif est de pouvoir détecter des ruptures plus générales.
  - Chap. 6 : Numerical experiments and evaluation. Ce chapitre contient les comparaisons expérimentales de méthodes de segmentation gloutonnes, avec des algorithmes standards.
- Part III : Supervised change point detection.
  - Chap. 7 : Calibrating the smoothing parameter through supervised learning. Un algorithme supervisé d'apprentissage de pénalité est présenté. Ici, le nombre de ruptures est inconnu.
  - Chap. 8 : Metric learning for change point detection. Une procédure supervisée pour calibrer la fonction de coût en utilisant des signaux annotés est présentée.
- Part IV : Statistical software.
  - Chap. 9 : `ruptures` : change point detection in Python. La librairie `ruptures` est décrite. La documentation détaillée est disponible en ligne, mais est également reproduite dans A : Documentation of `ruptures`. Ce chapitre peut être lu indépendamment.
- Appendices.
  - A : Documentation of `ruptures`. Une version est disponible en ligne[2].
  - B : An automated recording method in clinical consultation to rate the limp in lower limb osteoarthritis. Un travail publié, écrit dans le cadre de la collaboration avec Cognac-G, est inclus:
    - 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.
  - C : Template-based step detection from accelerometer signals. Un rapport technique est inclus. Il décrit un algorithme de détection de pas pour l'analyse de la marche.

[2]Disponible à ctruong.perso.math.cnrs.fr/ruptures

1 Introduction

Contents

1 Context of the thesis

General context. In the last decades, quantified-self applications have become more and more widespread in the population. More and more commercial solutions are available to compute health-related quantities such as the number of steps, the travelled distance, weight loss, energy spent, etc. This remarkable change the public's health habits is allowed by the stunning progression of sensors embedded in mobile devices. In the medical context, the same trend has been observed, thanks to the rise of cheap, wearable and easy to manipulate sensors. As a result, more and more physiological signals, such as electrocardiogram (ECG), electroencephalogram (EEG), body and limb accelerations, are being collected. There are many benefits to this trend: facilitating early diagnosis, promoting telemedicine, making healthcare more affordable, etc. Before reaping the fruits of this technological step forward, automatic and objective methods to extract information from this substantial volume of raw data must be designed. The transition from raw signals to actionable data is crucial in the success of next-generation healthcare and has become a real subject of interest.

Collaboration with Cognac-G. During my PhD, I have collaborated with Cognac-G, a research team regrouping machine learning researchers and medical researchers, gathered around the quantification of human and animal behaviour. To that end, several experimental protocols have been developed for a wide range of clinical problems from mice breathing or human locomotion to young infant eye movements. Each protocol is monitored with one or several sensors to provide an objective quantification of the phenomenon of interest. Resulting univariate or multivariate time series, known as physiological signals, are then studied. The first challenge consists in extracting the relevant information from these signals, in order to interpret them and to help understanding the physiological, biological or bio-mechanical mechanisms that produced them. The second challenge is to automatize the quantification process in order to provide tools that can be used by doctors for the longitudinal follow-up and the inter-individual comparison of their patients.

2 Motivations

2.1 From raw data to knowledge

In the vast majority of situations, the clinical context imposes practical constraints on the data acquisition process. For instance, clinicians might not have the possibility to repeatedly turn on and off the sensors at the beginning and end of the phenomena of interest. Also, subjects that are being monitored might be asked to perform successively different physical activities. In those situations, collected signals are made of consecutive phases, and precise information about the start and end of each phase is not always available. One crucial step to pre-process the large amount of such time series is called *signal segmentation* or *change point detection*. (Both terms are equivalently used in the remainder of the manuscript.) It consists in finding the temporal boundaries of the successive regimes of the signal, so that they can either be removed or further studied. This step is critical in the contextualization of long time series.

This setting is often encountered in daily clinical practice, when subjects can be monitored while they undergo a medical protocol. Typically, a clinician ask a patient to perform several consecutive physical exercises while some sensors record some physiological and bio-mechanical variables (e.g. heart rate, oxygen uptake, body acceleration). In order to quantify the evolution of the patient during the protocol, the monitoring signal is segmented, meaning that it is split in sub-signals, each corresponding to a coherent phase (for instance, a single exercise). Certain features of interest are then computed for each phase. This scheme is used for the study in human locomotion (more precisely, gait analysis) which is presented later in this manuscript and is illustrated on Figure 1.1.

From a practical standpoint, signal segmentation can be carried out manually by clinicians. For instance, they can record start and end timestamps of phenomena of interest, or mark *a posteriori* the time of changes by analysing the raw time series. However, both approaches can be cumbersome, if the protocol is complex and requires the full attention of the clinician, or if the changes are not easily visible from the raw signals. Indeed, even though trained specialists are able to assess a patient's state by eye, applying this expertise on time series is challenging. Also, in clinical trials, it can be critical to reduce the number of manual operations, which are open to interpretation and therefore subjective.

This motivates the study of automatic signal segmentation methods. To cope with the diversity of physiological time series, robust and versatile algorithms are needed, along with

systematic procedures to calibrate them. The objective is to capture the medical expertise of clinicians, with as little human intervention as possible. In addition, algorithms must bring little to no computational overhead if they are to be used in daily clinical routine.

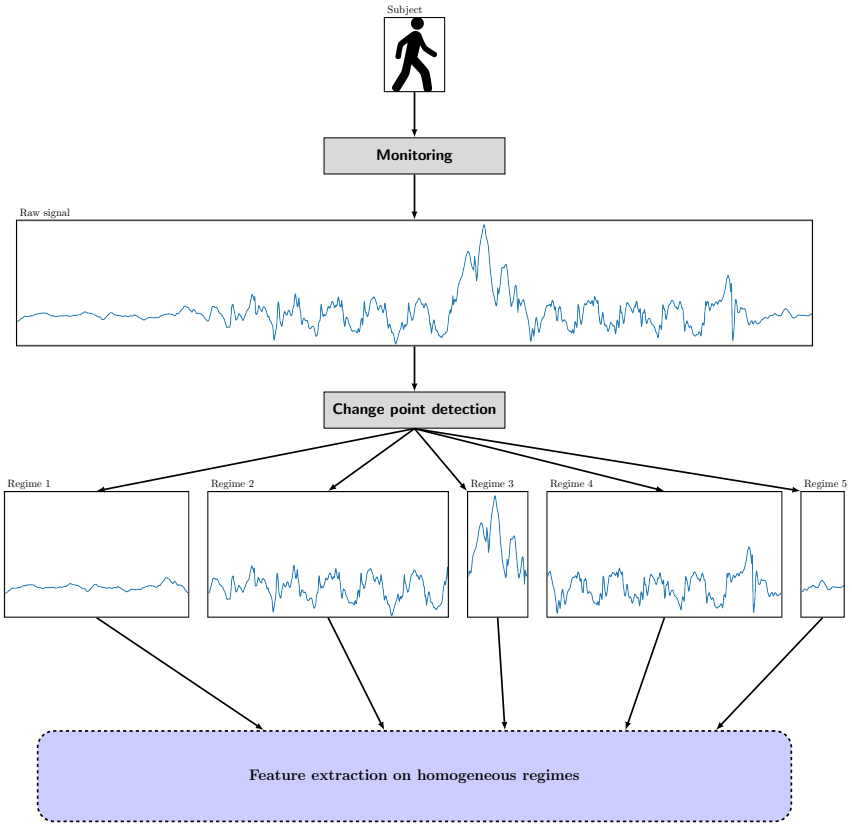


Figure 1.1: Flowchart of a study scheme, for gait analysis.

2.2 Motivating examples

We now describe three motivating examples, all originating from a cooperation between machine learning researchers and medical researchers, in Cognac-G. Special emphasis is put on the first example (analysis of human gait) because it was, at the beginning, the project at the most advanced stage, in terms of collected data, annotations and publications. It is present throughout this manuscript and is the touchstone of segmentation accuracy, for the contributions of this thesis.

2.2.1 Analysis of human gait

Context. The human motion results from a complex process, which requires the coordination of many muscles. Certain pathologies (such as Parkinson's disease, arthritis, stroke, obesity, diabetes,...) may alter the locomotion, threatening the autonomy of patients and increasing the risk of fall. The objective quantification and assessment of locomotion is therefore a crucial problem, that has been addressed in the literature by measuring the movement with several types of sensors such as inertial sensors, instrumented mat, force

platforms, camera-optical tracking system or force-sensitive resistors insoles. The signals obtained from these sensors are processed (automatically or manually) so as to extract some features that characterize the locomotion (speed, variability, smoothness,...).

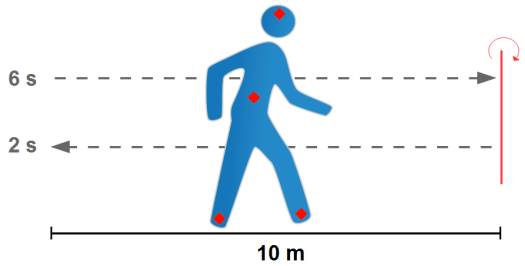
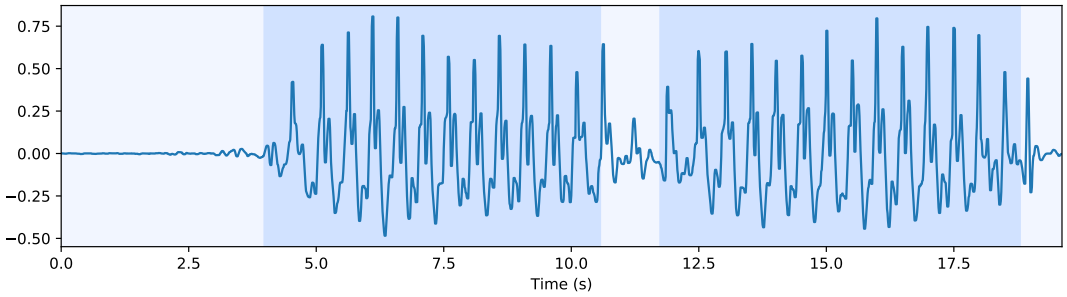
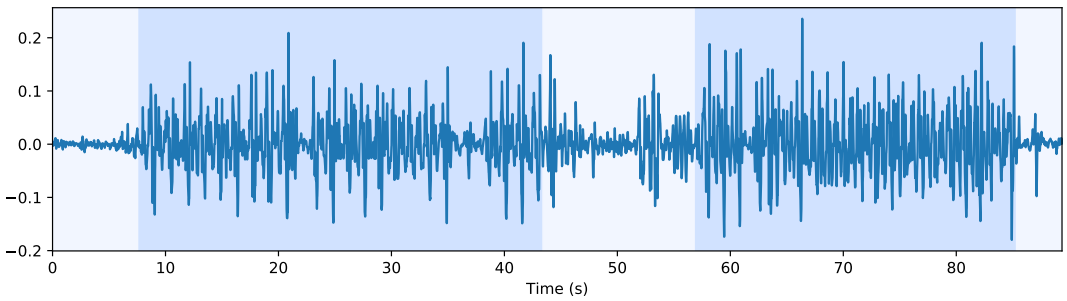


Figure 1.2: Scheme of the protocol used for the analysis of the human gait. Red dots indicate sensor positions.



(a) Healthy control subject.



(b) Osteoarthritis patient.

Figure 1.3: Vertical acceleration (m/s^2) of the lower back sensor for two different subjects. Alternating colours mark the consecutive phases: “Stand”, “Walk”, “Turnaround”, “Walk” and “Stop”.

Protocol. In this context, a clinical protocol has been conceived and implemented within Cognac-G for the study of human locomotion using Inertial Measurement Units (IMUs) which are composed of 3D accelerometers, 3D gyroscopes and 3D magnetometers. The main advantages of these sensors is that they are relatively low-cost, they do not require a dedicated room for the experiments, and their small size make them easy to handle in day-to-day clinical situations. The data used for the conception and testing of the method presented in this thesis has been provided by the following medical departments:

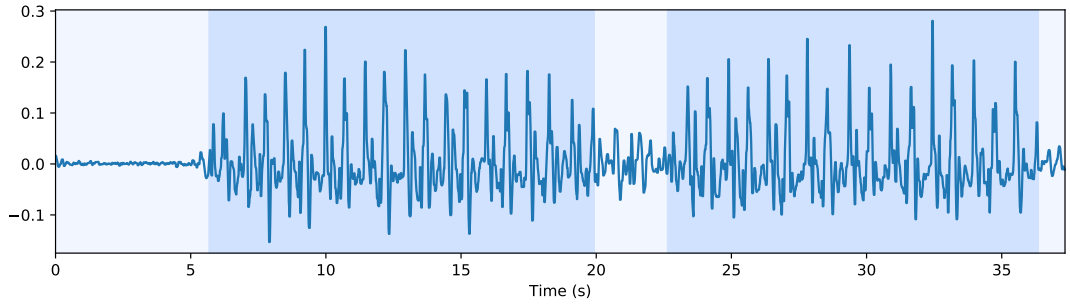
Service de chirurgie orthopédique et de traumatologie de l'Hôpital Européen Georges Pompidou, Assistance Publique des Hôpitaux de Paris, Service de médecine physique et de réadaptation de l'Hôpital Fernand Widal, Assistance Publique des Hôpitaux de Paris, Service de neurologie de l'Hôpital d'Instruction des Armées du Val de Grâce, Service de Santé des Armées. The study was validated by a local ethic committee and all subjects gave their written consent to participate. All signals have been acquired at 100 Hz with wireless XSens MTw^{TM} sensors located at lower back and fixed using a Velcro band designed by XSens^{TM}. All subjects were asked to stand still for 6 seconds, walk 10 meters at preferred walking speed on a level surface, turn around, walk back, stand still 2 seconds (illustrated on Figure 1.2). Two examples of recorded signals are displayed on Figure 1.3. The two flat parts at the extremities of the signal correspond to periods when the subject is standing still. The repeated patterns represent the footsteps. Depending on the pathology (or absence of), the length of the signal varies from 20 seconds to 90 seconds. Also, footsteps and the turnaround are less visible on the signal for the osteoarthritis patient.

Application of signal segmentation. Spectral properties of the collected signals can provide to the clinicians features adapted to gait analysis [26]. Indeed, as displayed on Figure 1.4, walking phases show a strong harmonic structure. However, when comparing the power spectral density (psd) of each "Walk" regime, differences with the psd of the whole signal are observed. This is due to the fact that the harmonic structure of the complete recording is corrupted by non-periodic phases, namely "Stand", "Turnaround" and "Stop". In addition, the two "Walk" regimes (forward and back) do not have the same frequency distribution. More precisely, the frequency peaks for the first and second "Walk" are respectively located around 2.6 Hz and 1.4 Hz. This frequency shift, which can serve as a measure of the subject's fatigue, can only be detected thanks to the segmentation of the signal.

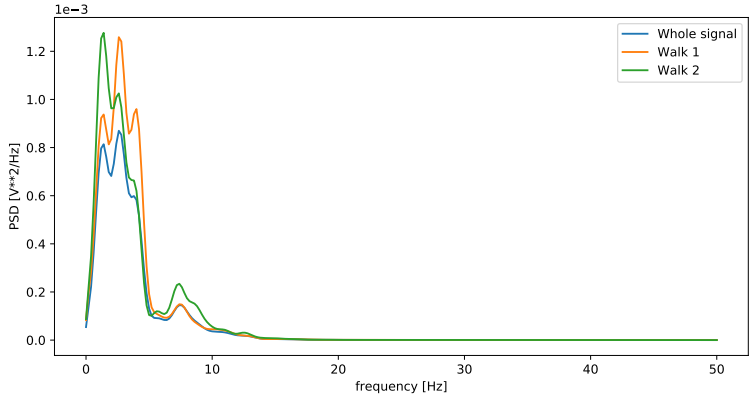
2.2.2 Other examples of physiological signals

Two other examples of collaborations within Cognac-G are presented. In both situations, signal segmentation is a crucial pre-processing step.

Analysis of mouse respiratory control. Acetylcholine (ACh) is a muscle-activating neurotransmitter (an organic chemical released by the nervous system to send signals) that participates in a large number of body functions, among which respiration. Cholinergic substances (meaning that they alter the release or activation capability of ACh) can be found in numerous drugs, toxins and chemical nerve agents, and cause respiratory failures. The objective quantification and assessment of breathing behaviours during cholinergic crisis can lead to a better understanding of the influence of ACh on respiratory control. In the literature, this issue has been addressed by monitoring mice after they have been exposed to specific gas mixtures.
In this context, a protocol has been conceived and implemented by researchers from Cognac-G, in accord with European Union and French Government animal protection laws. Several physiological variables related to breathing (such as the duration of inspiration and expiration, breathing frequency,...) are recorded in mice with particular deficits in cholinergic functions. Mice were placed in a plethysmograph chamber (a sealed chamber used to measure volume changes within the lungs) for 15-20 minutes, taken out to be



(a) Vertical acceleration of the lower back sensor.



(b) Power spectral density.

Figure 1.4: Signal example. (Top) Vertical acceleration (m/s^2) of the lower back sensor for a subject. Alternating colours mark the consecutive phases: “Stand”, “Walk”, “Turnaround”, “Walk” and “Stop”. (Bottom) Power spectral density for the whole signal, the first “Walk” phase and the second “Walk” phase.

exposed to gas and put back in the chamber. An illustrative example is displayed on Figure 1.5. The first period of the signal (the first 20 minutes) provides baseline values for the recorded variables. In the second period of the signal, different regimes can be observed, corresponding to different states of the mouse: calm in the beginning, then stressed with pronounced respiratory difficulties.

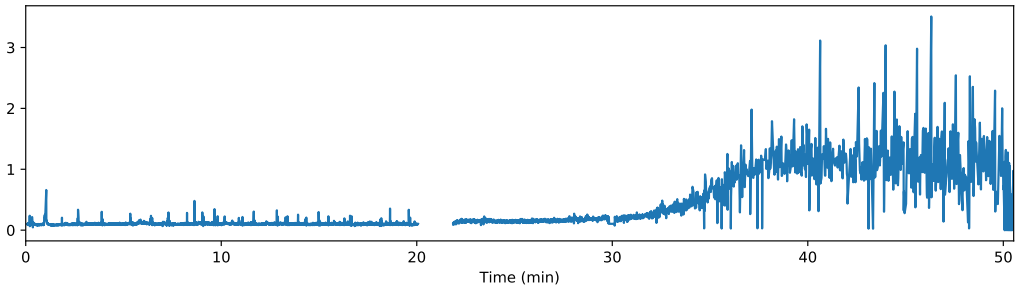


Figure 1.5: Evolution of the duration (in seconds) of expiration of a mouse (sampling frequency: 5 Hz). After 20 minutes, the mouse is exposed to a specific gas mixture. The gap in the signal corresponds to the manipulation of mouse.

Analysis of consciousness during anaesthesia. General anaesthesia consists in a medically induced coma, and is mandatory for certain surgeries. To reduce the risks associated with such a procedure, it is common practice to monitor the depth of consciousness of patients, with electroencephalogram (EEG) signals. The patient's state (awake, sedated,...) is then inferred using a suitable algorithm. The objective is to prevent any adverse event during and after the surgery and to reduce drug consumption. However, the analysis of recorded signals suffers several limitations, among which the sensitivity of the EEG to external electronic devices.
In this context, Cognac-G is developing a protocol for the monitoring of anaesthetized patients. It consists in several devices (that include an EEG, an electrocardiogram, a pulse oximeter, among other sensors) and algorithmic treatments of the collected signals. One of the issues this protocol aims at solving is the detection of artefacts from external electromagnetic field in the environment. An illustrative example is displayed on Figure 1.6. Several outliers and periods of high voltage amplitude can be seen in this recording. They are the consequence of electrosurgical devices that are used to cut and cauterize tissue. Removing such periods is a crucial step in order to infer the patient's state.

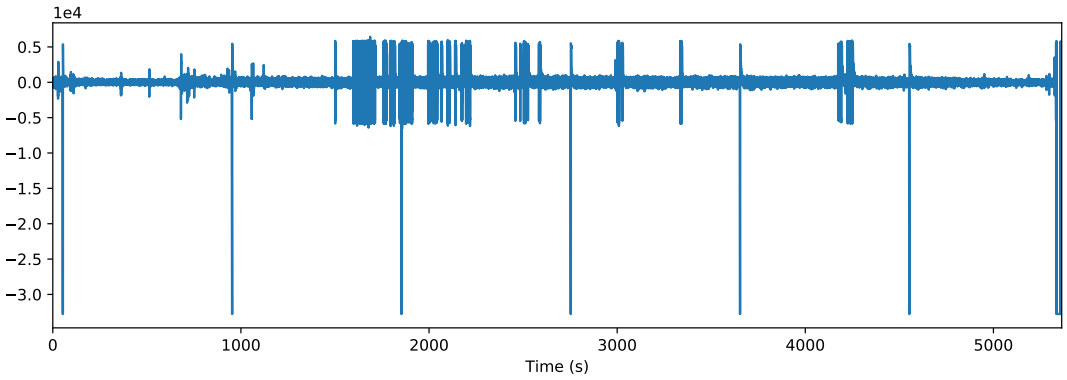


Figure 1.6: EEG recording (in μV) of a patient during anaesthesia (sampling frequency: 100 Hz).

3 Change point detection for physiological data

Several general principles have emerged from the previously described context. Those principles have heavily influenced the shape of change point detection algorithms proposed in this manuscript. In particular, they have lead us to consider strategies based on supervised learning.

3.1 General principles

We list the general principles that serve as guidelines for the algorithmic contributions of this thesis.

Computational cost and robustness. Monitoring a subject undergoing a specific protocol can result in a substantial volume of raw data. Indeed, a protocol can last for a significant period of time (in Cognac-G, they range from seconds to hours), the sensors

can have a high sampling frequency to capture a high frequency phenomenon (up to 1000 Hz in ophthalmic studies) or there can be many sensors collecting simultaneously several physiological characteristics. However, in routine clinical practice, clinicians expect the information extracted from the raw data to be available quickly, so that it can be viewed and interpreted within the consultation time. Also, for practical and legal reasons, the computational resources to run the algorithms are often limited to the clinicians' laptops or embedded devices. As a consequence, in order to process the raw data in a reasonable time, we have to consider sub-optimal segmentation methods, that impose little computational overhead. The trade-off is that such algorithms are less robust, comparatively to optimal methods. Therefore, in this thesis, we focus on providing a sub-optimal algorithm, that can accommodate time complexity constraints while retaining most of the robustness of optimal procedures.

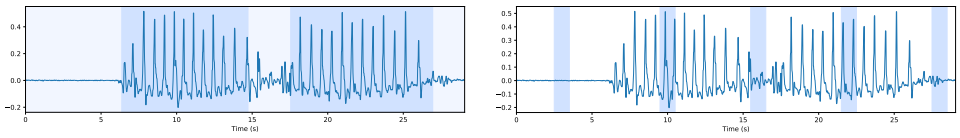
Versatility. Within the Cognac-G ecosystem, there is a great deal of variability in the type of signals that are collected. This is due to the fact that a large number of subjects have been monitored and several monitoring settings are used, resulting in a wide range of clinical parameter values (e.g. age, weight or pathology) and sensors. Depending on the context, signals can have very different dimensions, sampling rates, units, distributions, etc. Because of this variability, we can only consider a general class of segmentation methods that requires little to no assumption about the form of the underlying signal model or the location of change points. Also, those algorithms must be able to detect either a known or unknown number of changes, as both situations can arise in our context.

Automatic calibration. Ordinarily, collaborations within Cognac-G start with a long period of discussion between computer scientists and medical researchers. During this period, a suitable statistical model is chosen and calibrated to the data at hand. This process can be difficult and time-consuming, depending on the statistical education of the clinicians and the medical education of the statisticians. Indeed, even though clinicians have a deep understanding of the monitored physiological phenomenon, translating their medical expertise into statistical terms is a complex exercise. One of the objectives of this work is to move towards an automatization of this operation. More precisely, we aim at designing mechanisms to automatically calibrate segmentation methods, thus removing the need for a time-consuming hand-tuning of parameters (often done by trial and error).

3.2 Learning from experts

As a result of the general principles described above, we propose in this thesis to apply supervised learning procedures to change point detection. Generally speaking, such procedures can infer complex decision rules only using relevant examples. In our context, clinicians are able to provide such examples: they consist in signals, manually segmented by clinicians. Our objective is to use those examples to design a change point detection method able to replicate the segmentation strategy of the clinicians. Here, two types of annotations (or labels, in the supervised learning terminology) are considered: full and partial. For a *fully annotated signal*, the timestamps of all changes are provided by an expert. For a *partially annotated signal*, the expert only marks portions of the signal as homogeneous, meaning that they do not contain any change point. In this situation, exact locations of the changes are not known. The two types of annotations are illustrated on

Figure 1.7. Two use-case scenarios are commonly found in the context of Cognac-G. In the first scenario, a clinician provides one or several (fully or partially) annotated signals. The objective is to segment any new signal that comes from the same monitoring setting. To that end, a detection algorithm is calibrated, in a supervised fashion, using the (training) set of annotation signals, and then applied on new signals. In the second scenario, a clinician provides one or several partially annotated signals, that might not come from the same monitoring settings. The objective is to complete the partial segmentations of those signals, i.e. recover all change points. Here, a detection algorithm is calibrated, in a supervised fashion, on each (partially annotated) signal, and then applied on the same signal.



(a) Full annotation (b) Partial annotation

Figure 1.7: Two types of signal annotation (for a signal example, used for gait analysis). (Left) Alternating colours mark consecutive regimes. (Right) Blue areas denote portions of signals considered as homogeneous (i.e. not containing any change point).

4 Change point detection framework

In this thesis we focus on the change point detection task. We propose a general formulation in which we consider a multivariate non-stationary random process $y = \{y_1, \dots, y_T\}$ that takes value in $\mathbb{R}^d$ ($d \geq 1$). The signal y is assumed to be piecewise stationary, meaning that some characteristics of the process change abruptly at some unknown instants $t_1^\star < t_2^\star < \cdots < t_K^\star$. Change point detection consists in estimating the indexes $t_k^\star$. Depending on the context, the number K of changes may be known or unknown, in which case it has to be estimated too. It is important to note that, in our context, both situations are equally important. Indeed, the number of changes is often determined by the monitoring protocol. Also, in the applications that motivated this thesis in the first place, signal analysis is always performed a posteriori, therefore we focus on offline (also retrospective or a posteriori) change point detection. Conversely, the so-called on-line setting, in which the signal samples are assumed to be revealed progressively, was originally introduced for real-time signal analysis and is beyond our scope.

Change point detection literature is rich and applications are numerous. The first works on this subject go back to the 50s [130, 131]: the goal was to locate a shift in the mean of independent and identically distributed (i.i.d.) Gaussian variables. The main application of those methods was industrial quality control. Since then, this problem has been actively investigated, resulting in a rich and diverse literature. We give references to comprehensive review books that are dedicated exclusively to change point detection and cover a large span of this research area, from a parametric and non-parametric standpoints [29, 36, 45, 52, 102]. (More references can be found in the following review chapter.)

4.1 Generic detection methods

Parts of our contributions cover the choice and calibration of change point detection methods. The objective is to make this crucial step automatic for a practitioner who is not an expert in non-stationary time series modelling. To that end, we make necessary methodological assumptions on the shape of change detection methods. This defines a framework, which encompasses many settings from the literature and has explicitly been used in several practical contributions [98, 121, 139] as well as theoretical [107, 110].

Notations. In the remainder of the chapter, we use the following notations. For a given signal $y = \{y_t\}_{t=1}^{T}$, the $(b-a)$-sample long sub-signal $\{y_t\}_{t=a+1}^{b}$ $(1 \leq a < b \leq T)$ is simply denoted $y_{a..b}$; the complete signal is therefore $y = y_{0..T}$. A set of indexes is denoted by a calligraphic letter: $\mathcal{T} = \{t_1, t_2, \dots\} \subset \{1, \dots, T\}$, and its cardinal is $|\mathcal{T}|$. For a set of indexes $\mathcal{T} = \{t_1, \dots, t_K\}$, the dummy indexes $t_0 := 0$ and $t_{K+1} := T$ are implicitly available.

Methodological strategy. Change point detection is a modelling tool for analysis of non-stationary time series using simple models on each sub-segment. The goal is to find periods of homogeneity in the behaviour of the signal, or equivalently, to find moments of change. When cast as a model selection problem, change point detection amounts to choosing the best possible segmentation $\mathcal{T}$ according to a quantitative criterion $V(\mathcal{T}, y)$ that must be minimized. (The function $V(\mathcal{T}, y)$ is simply denoted $V(\mathcal{T})$ when it is obvious from the context that it refers to the signal y.) The choice of the criterion function $V(\cdot)$ depends on preliminary knowledge on the task at hand. In this work, we make the assumption that the criterion function $V(\mathcal{T})$ for a particular segmentation is a sum of costs of all the segments that define the segmentation:

$$V(\mathcal{T}, y) := \sum_{k=0}^{K} c(y_{t_k..t_{k+1}}) \tag{1.1}$$

where $c(\cdot)$ is a cost function which measures goodness-of-fit of the sub-signal $y_{t_k..t_{k+1}} = \{y_t\}_{t_k+1}^{t_{k+1}}$ to a specific model. The "best segmentation" $\widehat{\mathcal{T}}$ is the minimizer of the criterion $V(\mathcal{T})$. In practice, depending on whether the number K of change points is known beforehand, change point detection methods fall into two categories. If K is known (based on preliminary knowledge), the change point estimate $\widehat{\mathcal{T}}$ is the minimizer of the discrete optimization problem

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}). \tag{1.2}$$

If K is not known and has to be estimated, the change point estimation $\widehat{\mathcal{T}}$ is the minimizer of the penalized discrete optimization problem

$$\min_{\mathcal{T}} V(\mathcal{T}) + \text{pen}(\mathcal{T}) \tag{1.3}$$

where $\text{pen}(\mathcal{T})$ is an appropriate measure of the complexity of a segmentation $\mathcal{T}$. All change point detection methods considered in this work yield an exact or an approximate solution to either (1.2) or (1.3), with the function $V(\mathcal{T}, y)$ adhering to the format (1.1).

Examples. Examples of procedures that follow this format include maximum likelihood estimation (MLE) and piecewise regression. In the context of MLE, the signal samples are assumed to follow a process distribution $P(\cdot|\theta)$ parametrized by θ. Between two change points, the parameter θ is constant, and there is no inter-segment dependence. Here, we would like to detect when the value of parameter θ changes. MLE carried out by solving either the formulations (1.2) or (1.3), upon setting the cost function $c(\cdot)$ to be equal to the minimum negative log likelihood function, meaning that

$$c(y_{a..b}) := \min_{\theta} [-\log P(y_{a..b}|\theta)]. \tag{1.4}$$

In (parametric or non-parametric) regression analysis, the signal y is called a response variable, and we consider a signal $x = \{x_t\}_{t=1}^{T}$ of explanatory variables: the objective is to find a predictor function f within a subset of a functional space (for instance, linear functions, polynomial functions, etc.) such that the squared residuals $\|y_t - f(x_t)\|^2$ are as small as possible. In the piecewise regression setting, the predictor can change from one segment to the next. To find the associated change points, several regression tasks are separately performed on each segment. Upon setting the cost function $c(\cdot)$ equal to the sum of squared residuals

$$c(y_{a..b}) := \min_{f} \left[\sum_{t=a+1}^{b} \|y_t - f(x_t)\|^2 \right], \tag{1.5}$$

the change point detection problems (1.2) or (1.3) are equivalent to piecewise regression tasks.

Limitations. The described framework, however general, is introduced to tackle issues encountered in Cognac-G. Certain approaches do not fit in this framework. In particular, Bayesian approaches are not considered in the remainder ofthis thesis, even though they provide state-of-the-art results in several domains, such as speech and sound processing. The most well-known Bayesian algorithm is the Hidden Markov Model (HMM) [138]. This model was later extended, for instance with Dirichlet processes [99, 123] or product partition models [27, 28]. A common feature of Bayesian methods is the need to specify a prior on the location of change points. However, in the context of Cognac-G, such priors have not proven to be experimentally justified. The interested reader can find reviews of Bayesian approaches in [36] and [45].

4.2 Designing detection methods

The previous assumptions have an important consequence on the structure of methods within the adopted framework: change point detection methods which adhere to the format (1.2) or (1.3) are characterized by three elements.

Type of changes. The type of change a method is able to detect is encoded by its *cost function* $c(\cdot)$. For instance, setting the cost function as in (1.4) allows a method to detect changes in the value of the parameter θ of the process distribution $P(\cdot|\theta)$. Finding the relevant cost function for a given task is a critical part of the analysis a change point detection expert has to carry out.

Number of changes. The number of changes to detect can either be known (through *a priori* knowledge) or unknown. When it is unknown, it must be estimated along with the change point locations. To that end, a *complexity penalty* $\text{pen}(\cdot)$ (1.3) is introduced to balance out the goodness-of-fit term $V(\mathcal{T}, y)$. Complexity can be for instance proportional to the number of change points, in which case the penalty is called "linear". The choice of the complexity penalty is related to the amplitude of the changes to detect: with too "small" a penalty (compared to the goodness-of-fit) in (1.3), many change points are detected, even those that are the result of noise. Conversely, too much penalization only detects the most significant changes, or even none.

Computational complexity. The last brick of a detection procedure is the *search method*: it is the resolution procedure for the optimization problems (1.2) and (1.3). In (1.2), minimization is performed over the set $\{\mathcal{T} \text{ s.t. } |\mathcal{T}| = K\}$ which contains $\binom{T-1}{K-1}$ elements. In (1.3), minimization is performed over the set $\{\mathcal{T} \text{ s.t. } 1 \leq |\mathcal{T}| < T\}$ which contains $\sum_{K=1}^{T-1} \binom{T-1}{K-1}$ elements. In both situations, exhaustive enumeration of all possible segmentations is impractical. The literature contains several methods to efficiently solve those problems, in an exact fashion [30, 98] or in an approximate fashion [29, 146]. Each method achieves a certain balance between computational complexity and accuracy. In this thesis, we focus on providing fast sub-optimal algorithm able to retain most of the robustness of optimal procedures.

5 Contributions

We summarize the contributions of this thesis as follows.

- **Greedy change point detection.** We develop a novel greedy search method for change detection that is a trade-off between accuracy and (execution) speed. It is a sequential approximation of exact detection methods (for a known and unknown number of changes) that can be combined with kernel-based cost functions. We prove its asymptotic consistency and show it is more accurate than standard approximate methods and faster than exact methods, on simulated and real-world signals.

- **Supervised change point detection.** The objective is to provide automatic procedures to find and calibrate change point detection methods for a given task. To that end, we develop supervised methods that rely on a training set of annotated signals, meaning that change point locations have been manually located beforehand.

    - **Calibrating the smoothing parameter through supervised learning.** This contribution covers one of the defining elements of detection methods: the tuning of the complexity penalty. An algorithm is developed to calibrate the smoothing parameter value for linear penalty functions in a supervised fashion. We use a convex formulation which remains valid for any cost functions, as opposed to other supervised methods from the literature.
    - **Metric learning.** This contribution deals with another component of detection methods: the cost function. A cost function, parametrized by a Mahalanobis-type pseudo-metric, is learned from a set of annotated signals. The chosen metric learning procedure can also be combined a kernel, to provide a non-linear

treatment of the signal samples. Once the cost function is learned, it can be used with any of the search methods and complexity penalties of the literature.

- **Literature review through a Python implementation.** We develop a Python library, called `ruptures`, dedicated to change point detection. It is implemented according to the previously introduced framework, and relies heavily on the literature review of this thesis. Cost functions, search methods and complexity constraints are implemented separately, so that all those elements can be combined and used to create a detection method. Thanks to its modular and consistent interface, many algorithms can be run by simply changing a few parameters.
  A complete documentation is available at ctruong.perso.math.cnrs.fr/ruptures. The code can be found online at reine.cmla.ens-cachan.fr[1].

6 Overview of the manuscript

The remainder of the manuscript is structured in four parts.

- Part I : Literature review and evaluation framework
  - Chap. 2 : A selective review of change point detection methods. This is a literature review in which state-of-the-art algorithms are described, from a computational and a theoretical point of view. Readers can find synthetic summary tables: Table 2.1 on page 63 and Table 2.2 on page 76
  - Chap. 3 : Evaluation framework: metrics and data sets. This chapter presents the evaluation framework in which the performances of detection methods are assessed. The data sets and metrics described in this chapter are used throughout this manuscript.

  Many notations and experimental settings are defined in these two chapters and used in the remainder of the manuscript.

- Part II : Greedy change point detection.
  - Chap. 4 : Greedy change point detection. A greedy change point detection method is described. It aims at providing a trade-off between accuracy and (execution) speed for the segmentation of gait signals.
  - Chap. 5 : Greedy kernel change point detection. A kernel-based extension of the previous algorithm is presented. The objective is to be able to detect more general change points.
  - Chap. 6 : Numerical experiments and evaluation. This chapter contains the experimental comparisons of greedy segmentation methods, with standard algorithms.

- Part III : Supervised change point detection.

[1] https://reine.cmla.ens-cachan.fr/c.truong/ruptures/repository/latest/archive.zip

  - Chap. 7 : Calibrating the smoothing parameter through supervised learning. A supervised penalty learning algorithm is presented. Here the number of change points is unknown.
  - Chap. 8 : Metric learning for change point detection. A supervised procedure to calibrate the cost function using annotated signals is presented.

- Part IV : Statistical software.
  - Chap. 9 : `ruptures` : change point detection in Python. The library `ruptures` is described. The detailed documentation is available online, but is also reproduced in A : Documentation of `ruptures`. This chapter can be read independently.

- Appendices.
  - A : Documentation of `ruptures`. A version is available online[2].
  - B : An automated recording method in clinical consultation to rate the limp in lower limb osteoarthritis. A published work, written as part of the collaboration with Cognac-G, is included:
    * 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.
  - C : Template-based step detection from accelerometer signals. A technical report is included. It describes a step detection algorithm for gait analysis.

7 Publications

- C. Truong, L. Oudre, and N. Vayatis. Segmentation de signaux physiologiques par optimisation globale. In *Proceedings of the Groupe de Recherche et d'Etudes en Traitement du Signal et des Images (GRETSI)*, Lyon, France, 2015

- L. Oudre, T. Moreau, and C. Truong. Détection de pas à partir de données d'accélérométrie. In *Proceedings of the Groupe de Recherche et d'Etudes en Traitement du Signal et des Images (GRETSI)*, Lyon, France, 2015

- R. Barrois-Müller, L. Oudre, T. Moreau, C. Truong, N. Vayatis, S. Buffat, A. Yelnik, C. de Waele, T. Gregory, S. Laporte, P. P. Vidal, and D. Ricard. Quantify osteoarthritis gait at the doctor's office: a simple pelvis accelerometer based method independent from footwear and aging. *Computer Methods in Biomechanics and Biomedical Engineering*, 18 Suppl 1:1880–1881, 2015

- L. Oudre, R. Barrois-Müller, T. Moreau, C. Truong, R. Dadashi, T. Grégory, D. Ricard, N. Vayatis, C. De Waele, A. Yelnik, and P.-P. Vidal. Détection automatique des pas à partir de capteurs inertiels pour la quantification de la marche en consultation. *Neurophysiologie Clinique/Clinical Neurophysiology*, 45(4-5):394, 2015

[2] Available at ctruong.perso.math.cnrs.fr/ruptures

- 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

- J. Audiffren, R. Barrois-Müller, C. Provost, É. Chiarovano, L. Oudre, T. Moreau, C. Truong, A. Yelnik, N. Vayatis, P.-P. Vidal, C. De Waele, S. Buffat, and D. Ricard. Évaluation de l'équilibre et prédiction des risques de chutes en utilisant une Wii board balance. *Neurophysiologie Clinique/Clinical Neurophysiology*, 45(4-5):403, 2015

- C. Truong, L. Oudre, and N. Vayatis. Penalty learning for changepoint detection. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017

- C. Truong, L. Oudre, and N. Vayatis. ruptures, change point detection in Python, 2018. URL http://mloss.org/software/view/700/

- C. Truong, L. Oudre, and N. Vayatis. ruptures: change point detection in Python. *ArXiv e-prints arXiv:1801.00826*, pages 1–5, 2018

- C. Truong, L. Oudre, and N. Vayatis. A review of change point detection. *arXiv preprint arXiv:1801.00718*, pages 1–31, 2018

- C. Truong, L. Oudre, and N. Vayatis. Greedy kernel change point detection with an application to physiological signals. *Submitted*, pages 1–5, 2018

Part I

Literature review and evaluation framework

2

A selective review of change point detection methods

Contents

Abstract

This chapter presents a review of change point detection methods that are adapted to the challenges of Cognac-G. A general yet structuring methodological strategy is adopted to organize this vast body of work. More precisely, detection algorithms are characterized by three elements: a cost function, a search method and a constraint on the number of changes. Each of those elements is reviewed separately. Implementation of the main algorithms, examples of applications and theoretical results are provided.

1 Framework of the thesis

The first works on change point detection go back to the 50s [130, 131]: the goal was to locate a shift in the mean of independent and identically distributed (i.i.d.) Gaussian variables. The main application was industrial quality control. Since then, this problem has been actively investigated, and is periodically the subject of in-depth monographs [29, 36, 45, 52]. Nevertheless, it is important to provide a review that is adapted to the challenges encountered in Cognac-G (see previous chapter). To that end, in this chapter, we propose a survey of algorithms for the offline detection of multiple change points in multivariate time series. Both parametric and non-parametric methods are presented. Practical implementations are also provided, for optimal approaches as well as fast heuristics that are able to handle large signals. Procedures to estimate the number of changes, when it is unknown, are described. The objective of this chapter is to facilitate the search of a suitable detection method for a given application. To that end, all reviewed detection algorithms are organized according to a comprehensive typology.

Notations. In the remainder of the chapter, we use the following notations. For a given signal $y = \{y_t\}_{t=1}^{T}$, the $(b-a)$-sample long sub-signal $\{y_t\}_{t=a+1}^{b}$ $(1 \leq a < b \leq T)$ is simply denoted $y_{a..b}$; the complete signal is therefore $y = y_{0..T}$. A set of indexes is denoted by a calligraphic letter: $\mathcal{T} = \{t_1, t_2, \dots\} \subset \{1, \dots, T\}$, and its cardinal is $|\mathcal{T}|$. For a set of indexes $\mathcal{T} = \{t_1, \dots, t_K\}$, the dummy indexes $t_0 := 0$ and $t_{K+1} := T$ are implicitly available.

1.1 Problem statement

In the remainder ofthis chapter, we consider a multivariate non-stationary random process $y = \{y_1, \dots, y_T\}$ that takes value in $\mathbb{R}^d$ $(d \geq 1)$. The signal y is assumed to be piecewise stationary, meaning that some characteristics of the process change abruptly at some unknown instants $t_1^\star < t_2^\star < \cdots < t_{K^\star}^\star$. Change point detection consists in estimating the indexes $t_k^\star$. Depending on the context, the number $K^\star$ of changes may or may not be known, in which case it has to be estimated too. It is important to note that, in the context of Cognac-G, both situations are equally important. We focus on offline (also retrospective or a posteriori) change point detection, in which segmentation is performed after the signal has been collected.

Formally, change point detection is cast as a model selection problem, which consists in choosing the best possible segmentation $\mathcal{T}$ according to a quantitative criterion $V(\mathcal{T}, y)$ that must be minimized. (The function $V(\mathcal{T}, y)$ is simply denoted $V(\mathcal{T})$ when it is obvious from the context that it refers to the signal y.) The choice of the criterion function $V(\cdot)$ depends on preliminary knowledge on the task at hand. In this work, we make the assumption that the criterion function $V(\mathcal{T})$ for a particular segmentation is a sum of costs of all the segments that define the segmentation:

$$V(\mathcal{T}, y) := \sum_{k=0}^{K} c(y_{t_k..t_{k+1}}) \tag{2.1}$$

where $c(\cdot)$ is a cost function which measures goodness-of-fit of the sub-signal $y_{t_k..t_{k+1}} = \{y_t\}_{t_k+1}^{t_{k+1}}$ to a specific model. The "best segmentation" $\widehat{\mathcal{T}}$ is the minimizer of the criterion

$V(\mathcal{T})$. In practice, depending on whether the number $K^\star$ of change points is known beforehand, change point detection methods fall into two categories.

Problem 1 The change point detection problem with a fixed number K of change points consists in solving the following discrete optimization problem

$$\min_{|\mathcal{T}|=K} V(\mathcal{T}). \tag{2.2}$$

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

$$\min_{\mathcal{T}} V(\mathcal{T}) + \text{pen}(\mathcal{T}) \tag{2.3}$$

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

All change point detection methods considered in this work yield an exact or an approximate solution to either Problem 1 or Problem 2, with the function $V(\mathcal{T}, y)$ adhering to the format (2.1).

1.2 Structure of change point detection methods

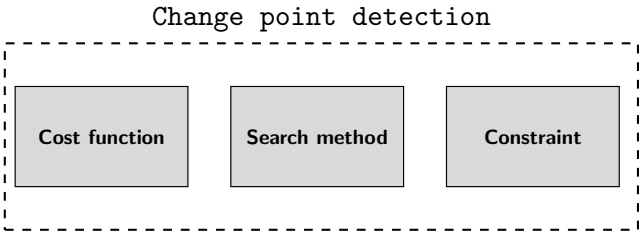


Figure 2.1: Typology of change point detection methods described in Chapter 2. Reviewed algorithms are defined by three elements: a cost function, a search method and a constraint (on the number of change points).

To better understand the strengths and weaknesses of change point detection methods, we propose to classify algorithms according to a comprehensive typology. Precisely, detection methods are expressed as the combination of the following three elements.

- **Cost function.** The cost function $c(\cdot)$ is a measure of "homogeneity". Its choice encodes the type of changes that can be detected. Intuitively, $c(y_{a..b})$ is expected to be low if the sub-signal $y_{a..b}$ is "homogeneous" (meaning that it does not contain any change point), and large if the sub-signal $y_{a..b}$ is "heterogeneous" (meaning that it contains one or several change points).
- **Search method.** The search method is the resolution procedure for the discrete optimization problems associated with Problem 1 and Problem 2. The literature contains several methods to efficiently solve those problems, in an exact fashion or in an approximate fashion. Each method strikes a balance between computational complexity and accuracy.

- **Constraint (on the number of change points).** When the number of changes is unknown (Problem 2), a constraint is added, in the form of a complexity penalty $\text{pen}(\cdot)$ (2.3), to balance out the goodness-of-fit term $V(\mathcal{T}, y)$. The choice of the complexity penalty is related to the amplitude of the changes to detect: with too "small" a penalty (compared to the goodness-of-fit) in (2.3), many change points are detected, even those that are the result of noise. Conversely, too much penalization only detects the most significant changes, or even none.

This typology of change point detection methods is schematically shown on Figure 2.1.

1.3 Asymptotic consistency

A natural question when designing detection algorithms is the consistency of estimated change point indexes, as the number of samples T goes to infinity. In the literature, the "asymptotic setting" is intuitively described as follows: the observed signal y is regarded as a realization of a continuous-time process on an equispaced grid of size $1/T$, and "T goes to infinity" means that the spacing of the sampling grid converges to 0. Precisely, for all $\tau \in [0,1]$, let $Y(\tau)$ denote an $\mathbb{R}^d$-valued random variable such that

$$y_t = Y(t/T) \quad \forall t = 1, \dots, T. \tag{2.4}$$

The continuous-time process undergoes $K^\star$ changes in the probability distribution at the time instants $\tau_k^\star \in (0,1)$. Those $\tau_k^\star$ are related to the change point indexes $t_k^\star$ through the following relationship:

$$t_k^\star = \lfloor T\tau_k^\star \rfloor. \tag{2.5}$$

Generally, for a given change point index t_k, the associated quantity $\tau_k = t_k/T \in (0,1)$ is referred to as a change point *fraction*. In particular, the change point fractions $\tau_k^\star$ ($k = 1, \dots, K^\star$) of the time-continuous process Y are change point indexes of the discrete-time signal y. Note that in this asymptotic setting, the lengths of each regime of y increase linearly with T. The notion of asymptotic consistency of a change point detection method is formally introduced as follows.

Definition 2.1 (Asymptotic consistency)**.** *A change point detection algorithm is said to be asymptotically consistent if the estimated segmentation $\widehat{\mathcal{T}} = \{\hat{t}_1, \hat{t}_2, \dots\}$ satisfies the following conditions, when $T \longrightarrow +\infty$:*

(i) $P(|\widehat{\mathcal{T}}| = K^\star) \longrightarrow 1$,

(ii) $\frac{1}{T}\left\|\widehat{\mathcal{T}} - \mathcal{T}^\star\right\|_\infty \xrightarrow{p} 0$,

where the distance between two change point sets is defined by

$$\left\|\widehat{\mathcal{T}} - \mathcal{T}^\star\right\|_\infty := \max\left\{ \max_{\hat{t}\in\widehat{\mathcal{T}}} \min_{t^\star\in\mathcal{T}^\star} |\hat{t} - t^\star|, \max_{t^\star\in\mathcal{T}^\star} \min_{\hat{t}\in\widehat{\mathcal{T}}} |\hat{t} - t^\star| \right\}. \tag{2.6}$$

Remark 2.1. *In Definition 2.1, the first condition is trivially verified when the number $K^\star$ of change points is known beforehand. As for the second condition, it implies that the estimated change point fractions are consistent, and not the indexes themselves. In general, distances $|\hat{t} - t^\star|$ between true change point indexes and their estimated counterparts do not converge to 0, even for simple models [19, 35, 40, 160]. As a result, consistency results in the literature only deal with change point fractions.*

1.4 Outline of this chapter

The organization of this review chapter reflects the typology of change point detection methods, which is schematically shown on Figure 2.1. Precisely, the three defining elements of a detection algorithm are reviewed separately. In Section 2, cost functions from the literature are presented, along with the associated signal model and the type of change that can be detected. Whenever possible, theoretical results on asymptotic consistency are also given. Section 3 lists search methods that efficiently solve the discrete optimizations associated with Problem 1 and Problem 2. Both exact and approximate methods are described. Constraints on the number of change points are reviewed in Section 4. A summary table of the literature review can be found in Section 5.

2 Models and cost functions

This section presents the first defining element of change detection methods, namely the cost function. In most cases, cost functions are derived from a signal model. In the following, models and their associated cost function are organized in two categories: parametric and non-parametric, as schematically shown in Figure 2.2. For each model, the most general formulation is first given, then special cases, if any, are described. A summary table of all reviewed costs can be found at the end of this section.

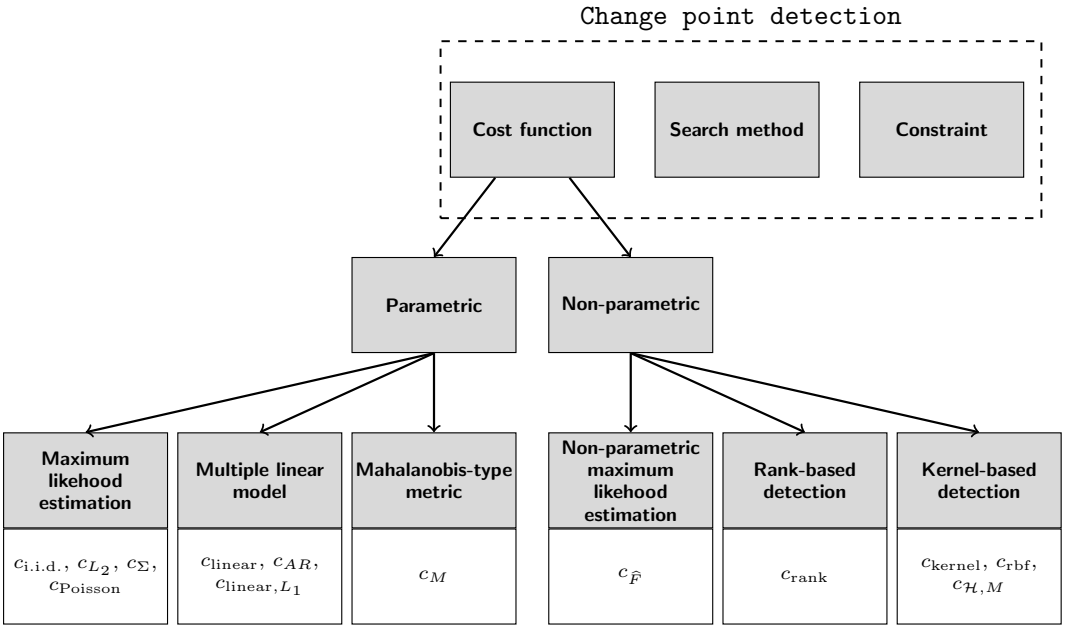


Figure 2.2: Typology of the cost functions described in Section 2.

2.1 Parametric models

Parametric detection methods focus on changes in a finite-dimensional parameter vector. Historically, they were the first to be introduced, and remain extensively studied in the literature.

2.1.1 Maximum likelihood estimation

Maximum likelihood procedures are ubiquitous in the change point detection literature. They generalize a large number of models and cost functions, such as mean-shifts and scale shifts in normally distributed data [107, 131], changes in the rate parameter of Poisson distributed data [99], etc. In the general setting of maximum likelihood estimation for change detection, the observed signal $y = \{y_1, \dots, y_T\}$ is composed of independent random variables, such that

$$y_t \sim \sum_{k=0}^{K^\star} f(\cdot|\theta_k)\mathbb{1}(t_k^\star < t \leq t_{k+1}^\star) \tag{M1}$$

where the $t_k^\star$ are change point indexes, the $f(\cdot|\theta)$ are probability density functions parametrized by the vector-valued parameter θ, and the θ_k are parameter values. In other words, the signal y is modelled by i.i.d. variables with piecewise constant distribution. The parameter θ represents a quantity of interest whose value changes abruptly at the unknown instants $t_k^\star$, which are to be estimated. Under this setting, change point detection is equivalent to maximum likelihood estimation if the sum of cost $V(\mathcal{T}, y)$ is equal to the negative log-likelihood. The corresponding cost function, denoted $c_{\text{i.i.d.}}$, is defined as follows.

Definition 2.2 ($c_{\text{i.i.d.}}$). *For a given parametric family of distribution densities $\{f(\cdot|\theta)|\theta \in \Theta\}$ where Θ is a compact subset of $\mathbb{R}^p$ (for a certain p), the cost function $c_{\text{i.i.d.}}$ is defined by*

$$c_{\text{i.i.d.}}(y_{a..b}) := -\sup_{\theta} \sum_{t=a+1}^{b} \log f(y_t|\theta). \tag{C1}$$

Model M1 and the related cost function $c_{\text{i.i.d.}}$ encompasses a large number of change point methods. Note that, in this context, the family of distributions must be known before performing the detection, usually thanks to prior knowledge on the data. Historically, the Gaussian distribution was first used, to model mean-shifts [102, 109, 146] and scale shifts [7, 99]. A large part of the literature then evolved towards other parametric distributions, most notably resorting to distributions from the general exponential family [62, 63, 120].
From a theoretical point of view, asymptotic consistency, as described in Definition 2.1, has been demonstrated, in the case of *a single change point*, first with Gaussian distribution (fixed variance), then for several specific distributions, e.g. Gaussian with mean and scale shifts [29, 45], discrete distributions [107], etc. The case with *multiple change points* has been tackled later. For certain distributions (e.g. Gaussian), the solutions of the change point detection problems (2.2) (known number of change points) and (2.3) (unknown number of change points) have been proven to be asymptotically consistent [1, 68]. The general case of multiple change points and a generic distribution family has been addressed decades after the change detection problem has been introduced: the solution of the change point detection problem with a known number of changes and a cost function set to $c_{\text{i.i.d.}}$ is asymptotically consistent [79]. This is true if certain assumptions are satisfied: (i) the signal follows the model (M1) for a distribution family that verifies some regularity assumptions (which are no different from the assumptions needed for generic maximum likelihood estimation, without any change point) and (ii) technical assumptions on the value of the cost function on homogeneous and heterogeneous sub-signals. As an example, distributions from the exponential family satisfy those assumptions.

Related cost functions. The general model (M1) has been applied with different families of distributions. We list below three notable examples and the associated cost functions: change in mean, change in mean and scale, and change in the rate parameter of count data.

- The mean-shift model is the earliest and one of the most studied model in the change point detection literature [47, 116, 122, 131, 146]. Here, the distribution is Gaussian, with fixed variance. In other words, the signal y is simply a sequence of independent normal random variables with piecewise constant mean and same variance. In this context, the cost function $c_{\text{i.i.d.}}$ becomes c_{L_2}, defined below. This cost function is also referred to as the quadratic error loss and has been applied for instance on DNA array data [63] and geology signals [43].

  Definition 2.3 (c_{L_2}). *The cost function c_{L_2} is given by*

  $$c_{L_2}(y_{a..b}) := \sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_2^2 \tag{C2}$$

  where $\bar{y}_{a..b}$ is the empirical mean of the sub-signal $y_{a..b}$.

- A natural extension to the mean-shift model consists in letting the variance abruptly change as well. In this context, the cost function $c_{\text{i.i.d.}}$ becomes c_{Σ}, defined below. This cost function can be used to detect changes in the first two moments of random (not necessarily Gaussian) variables, even though it is the Gaussian likelihood that is plugged in $c_{\text{i.i.d.}}$ [90, 107]. It has been applied for instance on stock market time series [107], biomedical data [43], and electric power consumption monitoring [80].

  Definition 2.4 (c_{Σ}). *The cost function c_{Σ} is given by*

  $$c_{\Sigma}(y_{a..b}) := (b-a)\log\det\widehat{\Sigma}_{a..b} + \sum_{t=a+1}^{b} (y_t - \bar{y}_{a..b})'\widehat{\Sigma}_{a..b}^{-1}(y_t - \bar{y}_{a..b}) \tag{C3}$$

  where $\bar{y}_{a..b}$ and $\widehat{\Sigma}_{a..b}$ are respectively the empirical mean and the empirical covariance matrix of the sub-signal $y_{a..b}$.

- Change point detection has also be applied on count data modelled by a Poisson distribution [48, 99]. More precisely, the signal y is a sequence of independent Poisson distributed random variables with piecewise constant rate parameter. In this context, the cost function $c_{\text{i.i.d.}}$ becomes c_{Poisson}, defined below.

  Definition 2.5 (c_{Poisson}). *The cost function $c_{Poisson}$ is given by*

  $$c_{Poisson}(y_{a..b}) := -(b-a)\bar{y}_{a..b}\log\bar{y}_{a..b} \tag{C4}$$

  where $\bar{y}_{a..b}$ is the empirical mean of the sub-signal $y_{a..b}$.

Remark 2.2. *A model slightly more general than (M1) can be formulated by letting the signal samples to be dependant and the distribution function $f(\cdot|\theta)$ to change over time. This can in particular model the presence of unwanted changes in the statistical properties of the signal*

(for instance in the statistical structure of the noise [107]). The function $f(\cdot|\theta)$ is replaced in (M1) by a sequence of distribution functions $f_t(\cdot|\theta)$ which are not assumed to be identical for all indexes t. Changes in the functions f_t are considered nuisance parameters and only the variations of the parameter θ must be detected. Properties on the asymptotic consistency of change point estimates can be obtained in this context. We refer the reader to [107, 108] for theoretical results.

2.1.2 Piecewise linear regression

Piecewise linear models are often found, most notably in the econometrics literature, to detect so-called "structural changes" [8–10]. In this context, a linear relationship between a response variable and covariates exists, and this relationship changes abruptly at some unknown instants. Formally, the observed signal y follows a piecewise linear model with change points located at the $t_k^\star$:

$$\forall\, t,\ t_k^\star < t \leq t_{k+1}^\star, \quad y_t = x_t' u_k + z_t' v + \varepsilon_t \quad (k = 0, \dots, K^\star) \tag{M2}$$

where the $u_k \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$ are unknown regression parameters and ε_t is noise. Under this setting, the observed signal y is regarded as a univariate response variable (i.e. $d = 1$) and the signals $x = \{x_t\}_{t=1}^T$ and $z = \{z_t\}_{t=1}^T$ are observed covariates, respectively $\mathbb{R}^p$-valued and $\mathbb{R}^q$-valued. In this context, change point detection can be carried out by fitting a linear regression on each segment of the signal. To that end, the sum of costs is made equal to the sum of squared residuals. The corresponding cost function, denoted c_{linear}, is defined as follows.

Definition 2.6 (c_{linear})**.** *For a signal y (response variable) and covariates x and z, the cost function c_{linear} is defined by*

$$c_{linear}(y_{a..b}) := \min_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} \sum_{t=a+1}^{b} (y_t - x_t' u - z_t' v)^2. \tag{C5}$$

In the literature, Model (M2) is also known as a *partial* structural change model because the linear relationship between y and x changes abruptly, while the linear relationship between y and z remains constant. The *pure* structural change model is obtained by removing the term $z_t' v$ from (M2). This formulation generalizes several well-known models such as the autoregressive (AR) model [3, 15], multiple regressions [10, 12], etc. A more general formulation of (M2) that can accommodate a multivariate response variable y exists [137], but is more involved, from a notational standpoint.
From a theoretical point of view, piecewise linear models are extensively studied in the context of change point detection by a series of important contributions [8–10, 12–18, 18, 132]. When the number of changes is known, the most general consistency result can be found in [17]. A multivariate extension of this result has been demonstrated in [137]. As for the more difficult situation of an unknown number of changes, statistical tests have been proposed for a single change [21] and multiple changes [14]. All of those results are obtained under various sets of general assumptions on the distributions of the covariates and the noise. The most general of those sets can be found in [133]. Roughly, in addition to some technical assumptions, it imposes the processes x and z to be weakly stationary within each regime, and precludes the noise process to have a unit root.

Related cost functions. In the rich literature related to piecewise linear models, the cost function c_{linear} has been applied and extended in several different settings. Two related cost functions are listed below.

- The first one is c_{linear,L_1}, which was introduced in order to accommodate certain noise distributions with heavy tails [9, 13] and is defined as follows.

  Definition 2.7 (c_{linear,L_1}). *For a signal y (response variable) and covariates x and z, the cost function c_{linear,L_1} is defined by*

  $$c_{linear,L_1}(y_{a..b}) := \min_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \sum_{t=a+1}^{b} |y_t - x_t'u - z_t'v|. \tag{C6}$$

  The difference between c_{linear,L_1} and c_{linear} lies in the norm used to measure errors: c_{linear,L_1} is based on a least absolute deviations criterion, while c_{linear} is based on a least squares criterion. As a result, c_{linear,L_1} is often applied on data with noise distributions with heavy tails [90, 120]. In practice, the cost function c_{linear,L_1} is computationally less efficient than the cost function c_{linear}, because the associated minimization problem (C6) has no analytical solution. Nevertheless, the cost function c_{linear,L_1} is often applied on economic and financial data [8–10]. For instance, changes in several economic parameters of the G-7 growth have been investigated using a piecewise linear model and c_{linear,L_1} [58].

- The second cost function related to c_{linear} has been introduced to deal with piecewise autoregressive signals. The autoregressive model is a popular representation of random processes, where each variable depends linearly on the previous variables. The associated cost function, denoted c_{AR}, is defined as follows.

  Definition 2.8 (c_{AR}). *For a signal y and an order $p \geq 1$, the cost function c_{AR} is defined by*

  $$c_{AR}(y_{a..b}) := \min_{u\in\mathbb{R}^p} \sum_{t=a+1}^{b} \left\|y_t - x_t'u\right\|^2 \tag{C7}$$

  where $x_t := [y_{t-1}, y_{t-2}, \ldots, y_{t-p}]$ is the vector of lagged samples.

  The piecewise autoregressive model is a special case of the generic piecewise linear model, where the term $z_t'v$ is removed (yielding a pure structural change model) and the covariate signal x is equal to the signal of lagged samples. The resulting cost function c_{AR} is able to detect shifts in the autoregressive coefficients of a non-stationary process [15, 41]. This model has been applied on EEG/ECG time series [137], functional magnetic resonance imaging (fMRI) time series [124] and speech recognition tasks [3].

2.1.3 Mahalanobis-type metric

The cost function c_{L_2} (C2), adapted for mean-shift detection, can be extended through the use of Mahalanobis-type pseudo-norm. Formally, for any symmetric positive semi-definite matrix $M \in \mathbb{R}^{d\times d}$, the associated pseudo-norm $\|\cdot\|_M$ is given by:

$$\|y_t\|_M^2 := y_t' M y_t \tag{2.7}$$

for any sample y_t. The resulting cost function c_M is defined as follows.

Definition 2.9 (c_M)**.** *The cost function c_M, parametrized by a symmetric positive semi-definite matrix $M \in \mathbb{R}^{d\times d}$, is given by*

$$c_M(y_{a..b}) := \sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_M^2 \tag{C8}$$

where $\bar{y}_{a..b}$ is the empirical mean of the sub-signal $y_{a..b}$.

Intuitively, measuring distances with the pseudo-norm $\|\cdot\|_M$ is equivalent to applying a linear transformation on the data and using the regular (Euclidean) norm $\|\cdot\|$. Indeed, decomposing the matrix $M = U'U$ yields:

$$\|y_t - y_s\|_M^2 = \|Uy_t - Uy_s\|^2. \tag{2.8}$$

Originally, the metric matrix M was set equal to the inverse of the covariance matrix, yielding the Mahalanobis metric [119], i.e.

$$M = \widehat{\Sigma}^{-1} \tag{2.9}$$

where $\widehat{\Sigma}$ is the empirical covariance matrix of the signal y. By using c_M, shifts in the mean of the transformed signal can be detected. In practice, the transformation U (or equivalently, the matrix M) is chosen to highlight relevant changes. This cost function generalizes all linear transformations of the data samples. In the context of change point detection, most of the transformations are unsupervised, for instance principal component analysis or linear discriminant analysis [76]. Supervised strategies are more rarely found, even though there exist numerous methods to learn a task-specific matrix M in the context of supervised classification [55, 76, 165]. Those strategies fall under the umbrella of metric learning algorithms. In the change point detection literature, there is only one work that proposes a supervised procedure to calibrate a metric matrix M [105]. In this contribution, the authors use a training set of annotated signals (meaning that an expert has provided the change point locations) to learn M iteratively. Roughly, at each step, a new matrix M is generated in order to improve change point detection accuracy on the training signals. However, using the cost function c_M is not adapted to certain applications, where a linear treatment of the data is insufficient. In that situation, a well-chosen non-linear transformation of the data samples must be applied beforehand [105].

2.2 Non-parametric models

When the assumptions of parametric models are not adapted to the data at hand, non-parametric change point detection methods can be more robust. Three major approaches are presented here, each based on different non-parametric statistics, such as the empirical cumulative distribution function, rank statistics and kernel estimation.

Signal model. Assume that the observed signal $y = \{y_1, \dots, y_T\}$ is composed of independent random variables, such that

$$y_t \sim \sum_{k=0}^{K^\star} F_k \, \mathbb{1}(t_k^\star < t \leq t_{k+1}^\star) \tag{M3}$$

where the $t_k^\star$ are change point indexes and the F_k are cumulative distribution functions (c.d.f), not necessarily parametric as in (M1). Under this setting, the sub-signal $y_{t_k^\star..t_{k+1}^\star}$, bounded by two change points, is composed of i.i.d. variables with c.d.f. F_k. When the F_k belong to a known parametric distribution family, change point detection is performed with the MLE approach described in Section 2.1.1, which consists in applying the cost function $c_{\text{i.i.d.}}$. However, this approach is not possible when the distribution family is either non-parametric or not known beforehand.

2.2.1 Non-parametric maximum likelihood

The first non-parametric cost function example, denoted $c_{\widehat{F}}$, has been introduced for the *single* change point detection problem in [59] and extended for *multiple* change points in [171]. It relies on the empirical cumulative distribution function (c.d.f.), estimated on sub-signals. Formally, the signal is assumed to be univariate (i.e. $d = 1$) and the empirical c.d.f. on the sub-signal $y_{a..b}$ is given by

$$\forall u \in \mathbb{R}, \quad \widehat{F}_{a..b}(u) := \frac{1}{b-a}\left[\sum_{t=a+1}^{b} \mathbb{1}(y_t < u) + 0.5 \times \mathbb{1}(y_t = u)\right]. \tag{2.10}$$

In order to derive a log-likelihood function that does not depend on the probability distribution of the data, i.e. the $f(\cdot|\theta_k)$, the authors use the following fact: for a fixed $u \in \mathbb{R}$, the empirical c.d.f. $\widehat{F}$ of n i.i.d. random variables, distributed from a certain c.d.f. F is such that $n\widehat{F}(u) \sim \text{Binomial}(n, F(u))$ [171]. This observation, combined with careful summation over u, allows a distribution-free maximum likelihood estimation. The resulting cost function $c_{\widehat{F}}$ is defined as follows. Interestingly, this strategy was first introduced to design non-parametric two-sample statistical tests, which were experimentally shown to be more powerful than classical tests such as Kolmogorov-Smirnov and Cramér-von Mises [59, 169].

Definition 2.10 ($c_{\widehat{F}}$). *The cost function $c_{\widehat{F}}$ is given by*

$$c_{\widehat{F}}(y_{a..b}) := -(b-a)\sum_{u=1}^{T} \frac{\widehat{F}_{a..b}(u)\log\widehat{F}_{a..b}(u) + (1-\widehat{F}_{a..b}(u))\log(1-\widehat{F}_{a..b}(u))}{(u-0.5)(T-u+0.5)} \tag{C9}$$

where the empirical c.d.f. $\widehat{F}_{a..b}$ is defined by (2.10).

From a theoretical point of view, asymptotic consistency of change point estimates is verified, when the number of change points is either known or unknown [171]. However, solving either one of the detection problems can be computationally intensive, because calculating the value of the cost function $c_{\widehat{F}}$ on one sub-signal requires to sum T terms, where T is the signal length. As a result, the total complexity of change point detection is of the order of $\mathcal{O}(T^3)$ [171]. To cope with this computational burden, several preliminary steps are proposed. For instance, irrelevant change point indexes can be removed before performing the detection, thanks to a screening step [171]. Also, the cost function $c_{\widehat{F}}$ can be approximated, by summing, in (C9), over a few (carefully chosen) terms, instead of T terms originally [78]. Thanks to those implementation techniques, the cost function $c_{\widehat{F}}$ has been applied on DNA sequences [171] and heart-rate monitoring signals [78].

2.2.2 Rank-based detection

In statistical inference, a popular strategy to derive distribution-free statistics is to replaced the data samples by their ranks within the set of pooled observations [50, 64, 114]. In the context of change point detection, this strategy has first been applied to detect a *single* change point [114, 117], and then has been extended by [118] to find *multiple* change points. The associated cost function, denoted c_{rank}, is defined as follows. Formally, it relies on the centered $\mathbb{R}^d$-valued "rank signal" $r = \{r_t\}_{t=1}^T$, given by

$$r_{t,j} := \sum_{s=1}^{T} \mathbb{1}(y_{s,j} \leq y_{t,j}) \; - \; \frac{T+1}{2}, \quad \forall 1 \leq t \leq T, \, \forall 1 \leq j \leq d. \tag{2.11}$$

In other words, $r_{t,j}$ is the (centered) rank of the j^{th} coordinate of the t^{th} sample, i.e. $y_{t,j}$, among the $\{y_{1,j}, y_{2,j}, \ldots, y_{T,j}\}$.

Definition 2.11 (c_{rank})**.** *The cost function c_{rank} is given by*

$$c_{rank}(y_{a..b}) := -(b-a)\, \bar{r}'_{a..b}\, \widehat{\Sigma}_r^{-1}\, \bar{r}_{a..b} \tag{C10}$$

where the signal r is defined in (2.11) and $\widehat{\Sigma}_r \in \mathbb{R}^{d \times d}$ is the following matrix

$$\widehat{\Sigma}_r := \frac{1}{T} \sum_{t=1}^{T} (r_t + 1/2)'(r_t + 1/2). \tag{2.12}$$

Intuitively, c_{rank} measures changes in the joint behaviour of the marginal rank statistics of each coordinate, which are contained in r. One of the advantages of this cost function is that it is invariant under any monotonic transformation of the data. Several well-known statistical hypothesis testing procedures are based on this scheme, for instance the Wilcoxon-Mann-Whitney test [163], the Friedman test [113], the Kruskal-Wallis test [93], and several others [50, 64]. From a computational point of view, two steps must be performed before the change point detection: the calculation of the rank statistics, in $\mathcal{O}(dT \log T)$ operations, and the calculation of the matrix $\widehat{\Sigma}_r$, in $\mathcal{O}(d^2 T + d^3)$ operations. The resulting algorithm has been applied on DNA sequences [118] and network traffic data [114, 117].

2.2.3 Kernel-based detection

A kernel-based method has been proposed by [72] to perform change point detection in a non-parametric setting. To that end, the original signal y is mapped onto a reproducing Hilbert space (rkhs) $\mathcal{H}$ associated with a user-defined kernel function $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The mapping function $\phi : \mathbb{R}^d \to \mathcal{H}$ onto this rkhs is implicitly defined by $\phi(y_t) = k(y_t, \cdot) \in \mathcal{H}$, resulting in the following inner-product and norm:

$$\langle \phi(y_s) | \phi(y_t) \rangle_{\mathcal{H}} = k(y_s, y_t) \quad \text{and} \quad \|\phi(y_t)\|_{\mathcal{H}}^2 = k(y_t, y_t) \tag{2.13}$$

for any samples $y_s, y_t \in \mathbb{R}^d$. The associated cost function, denoted c_{kernel}, is defined as follows. This kernel-based mapping is central to many machine learning developments such as support vector machine or clustering [70, 142].

Definition 2.12 (c_{kernel}). *For a given kernel function $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the cost function c_{kernel} is given by*

$$c_{kernel}(y_{a..b}) := \sum_{t=a+1}^{b} \|\phi(y_t) - \bar{\mu}_{a..b}\|_{\mathcal{H}}^2 \tag{C11}$$

where $\bar{\mu}_{a..b} \in \mathcal{H}$ is the empirical mean of the embedded signal $\{\phi(y_t)\}_{t=a+1}^{b}$ and $\|\cdot\|_{\mathcal{H}}$ is defined in (2.13).

Remark 2.3 (Computing the cost function). *Thanks to the well-known "kernel trick", the explicit computation of the mapped data samples $\phi(y_t)$ is not required to calculate the cost function value [39]. Indeed, after simple algebraic manipulations, $c_{kernel}(y_{a..b})$ can be rewritten as follows:*

$$c_{kernel}(y_{a..b}) = \sum_{t=a+1}^{b} k(y_t, y_t) \; - \; \frac{1}{b-a} \sum_{s,t=a+1}^{b} k(y_s, y_t). \tag{2.14}$$

Remark 2.4 (Intuition behind the cost function). *Intuitively, the cost function c_{kernel} is able to detect mean-shifts in the transformed signal $\{\phi(y_t)\}_t$. Its use is motivated in the context of Model M3 by the fact that, under certain conditions on the kernel function, changes in the probability distribution coincide with mean-shifts in the transformed signal. This connection has been investigated in several works on kernel methods [70, 142, 147, 150]. Formally, let $\mathbb{P}$ denote a probability distribution defined over $\mathbb{R}^d$. Then there exists a unique element $\mu_{\mathbb{P}} \in \mathcal{H}$ [150], called the mean embedding (of $\mathbb{P}$), such that*

$$\mu_{\mathbb{P}} = \mathbb{E}_{X \sim \mathbb{P}} \left[\phi(X)\right]. \tag{2.15}$$

In addition, the mapping $\mathbb{P} \mapsto \mu_{\mathbb{P}}$ is injective (in which case the kernel is said to be characteristic*), meaning that*

$$\mu_{\mathbb{P}} = \mu_{\mathbb{Q}} \Longleftrightarrow \mathbb{P} = \mathbb{Q}, \tag{2.16}$$

where $\mathbb{Q}$ denotes a probability distribution defined over $\mathbb{R}^d$. In order to determine if a kernel is characteristic (and therefore, useful for change point detection), several conditions can be found in the literature [70, 142, 150]. For instance, if a kernel $k(\cdot,\cdot)$ is translation invariant, meaning that $k(y_s, y_t) = \psi(y_s - y_t)\ \forall s,t$, where ψ is a bounded continuous positive definite function on $\mathbb{R}^d$, then it is characteristic [150]. This condition is verified by the commonly used Gaussian kernel. As a consequence, two transformed samples $\phi(y_s)$ and $\phi(y_t)$ are distributed around the same *mean value if they belong to the same regime, and around* different *mean-values if they each belong to two consecutive regimes. To put it another way, a signal that follows* (M3) *is mapped by $\phi(\cdot)$ to a random signal with piecewise constant mean.*

From a theoretical point of view, asymptotic consistency of the change point estimates has been demonstrated for both a known and unknown number of change points in the recent work of [69]. This result, as well as an important oracle inequality on the sum of cost $V(\mathcal{T})$ [5], also holds in a non-asymptotic setting. In addition, kernel change point detection was experimentally shown to be competitive in many different settings, in an unsupervised manner and with very few parameters to manually calibrate. For instance, the cost function c_{kernel} was applied on the Brain-Computer Interface (BCI) data set [72],on a video time series segmentation task [5] and DNA sequences [39].

Related cost functions. The cost function c_{kernel} can be combined with any kernel to accommodate various types of data (not just $\mathbb{R}^d$-valued signals). Notable examples of kernel functions include [147]:

- The linear kernel $k(x,y) = \langle x|y\rangle$ with $x,y \in \mathbb{R}^d$.
- The polynomial kernel $k(x,y) = (\langle x|y\rangle + C)^{deg}$ with $x,y \in \mathbb{R}^d$, and C and deg are parameters.
- The Gaussian kernel $k(x,y) = \exp(-\gamma \|x-y\|^2)$ with $x,y \in \mathbb{R}^d$ and $\gamma > 0$ is the so-called bandwidth parameter.
- The χ^2-kernel $k(x,y) = \exp(-\gamma \sum_i [(x_i - y_i)^2/(x_i + y_i)])$ with $\gamma \in \mathbb{R}$ a parameter. It is often used for histogram data.

Arguably, the most commonly used kernels for numerical data are the linear kernel and the Gaussian kernel. When combined with the linear kernel, the cost function c_{kernel} is formally equivalent to c_{L_2}. As for the Gaussian kernel, the associated cost function, denoted c_{rbf}, is defined as follows.

Definition 2.13 (c_{rbf})**.** *The cost function c_{rbf} is given by*

$$c_{rbf}(y_{a..b}) := (b-a) \; - \; \frac{1}{b-a} \sum_{s,t=a+1}^{b} \exp(-\gamma \|y_s - y_t\|^2) \tag{C12}$$

where $\gamma > 0$ is the so-called bandwidth parameter.

The parametric cost function c_M (based on a Mahalanobis-type norm) can be extended to the non-parametric setting through the use of a kernel. Formally, the Mahalanobis-type norm $\|\cdot\|_{\mathcal{H},M}$ in the feature space $\mathcal{H}$ is defined by

$$\|\phi(y_s) - \phi(y_t)\|^2_{\mathcal{H},M} = (\phi(y_s) - \phi(y_t))' M (\phi(y_s) - \phi(y_t)) \tag{2.17}$$

where M is a (possibly infinite dimensional) symmetric positive semi-definite matrix defined on $\mathcal{H}$. The associated cost function, denoted $c_{\mathcal{H},M}$, is defined below. Intuitively, using $c_{\mathcal{H},M}$ instead of c_M introduces a non-linear treatment of the data samples.

Definition 2.14 ($c_{\mathcal{H},M}$)**.** *For a given kernel function $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and M a symmetric positive semi-definite matrix defined on the associated rkhs $\mathcal{H}$, the cost function $c_{\mathcal{H},M}$ is given by*

$$c_{\mathcal{H},M}(y_{a..b}) := \sum_{t=a+1}^{b} \|\phi(y_t) - \bar{\mu}_{a..b}\|^2_{\mathcal{H},M} \tag{C13}$$

where $\mu_{a..b}$ is the empirical mean of the transformed sub-signal $\{\phi(y_t)\}_{t=a+1}^{b}$ and $\|\cdot\|_{\mathcal{H},M}$ is defined in (2.17).

2.3 Summary table

Reviewed cost functions (parametric and non-parametric) are summarized in Table 2.1. For each cost, the name, expression and parameters of interest are given.

Name	$c(y_{a..b})$	Parameters
$c_{\text{i.i.d.}}$ (C1)	$-\sup_\theta \sum_{t=a+1}^{b} \log f(y_t\|\theta)$	θ: changing parameter; density function: $f(\cdot\|\theta)$
c_{L_2} (C2)	$\sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_2^2$	$\bar{y}_{a..b}$: empirical mean of $y_{a..b}$
c_Σ (C3)	$(b-a)\log\det\widehat{\Sigma}_{a..b} + \sum_{t=a+1}^{b}(y_t - \bar{y}_{a..b})'\widehat{\Sigma}_{a..b}^{-1}(y_t - \bar{y}_{a..b})$	$\widehat{\Sigma}_{a..b}$: empirical covariance of $y_{a..b}$
c_{Poisson} (C4)	$-(b-a)\bar{y}_{a..b}\log\bar{y}_{a..b}$	$\bar{y}_{a..b}$: empirical mean of $y_{a..b}$
c_{linear} (C5)	$\min_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \sum_{t=a+1}^{b}(y_t - x_t'u - z_t'v)^2$	$x_t \in \mathbb{R}^p, z_t \in \mathbb{R}^q$: covariates
c_{linear,L_1} (C6)	$\min_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \sum_{t=a+1}^{b} \|y_t - x_t'u - z_t'v\|$	$x_t \in \mathbb{R}^p, z_t \in \mathbb{R}^q$: covariates
c_{AR} (C7)	$\min_{u\in\mathbb{R}^p} \sum_{t=a+1}^{b}(y_t - x_t'u)^2$	$x_t = [y_{t-1}, y_{t-2}, \ldots, y_{t-p}]$: lagged samples
c_M (C8)	$\sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_M^2$	$M \in \mathbb{R}^{d\times d}$: positive semi-definite matrix
$c_{\widehat{F}}$ (C9)	$-(b-a)\sum_{u=1}^{T} \frac{\widehat{F}_{a..b}(u)\log\widehat{F}_{a..b}(u) + (1-\widehat{F}_{a..b}(u))\log(1-\widehat{F}_{a..b}(u))}{(u-0.5)(T-u+0.5)}$	$\widehat{F}$: empirical c.d.f. (2.10)
c_{rank} (C10)	$-(b-a)\,\bar{r}_{a..b}'\,\widehat{\Sigma}_r^{-1}\,\bar{r}_{a..b}$	r: rank signal (2.11); $\widehat{\Sigma}_r$: empirical covariance of r (2.12)
c_{kernel} (C11)	$\sum_{t=a+1}^{b} k(y_t, y_t) - \frac{1}{b-a}\sum_{s,t=a+1}^{b} k(y_s, y_t)$	$k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$: kernel function
c_{rbf} (C12)	$(b-a) - \frac{1}{b-a}\sum_{s,t=a+1}^{b} \exp(-\gamma\|y_s - y_t\|^2)$	$\gamma > 0$: bandwidth parameter
$c_{\mathcal{H},M}$ (C13)	$\sum_{t=a+1}^{b} \|y_t - \bar{y}_{a..b}\|_{\mathcal{H},M}^2$	M: positive semi-definite matrix (in the feature space $\mathcal{H}$)

Table 2.1: Summary of cost reviewed functions

3 Search methods

This section presents the second defining element of change detection methods, namely the search method. Reviewed search methods are organized in two general categories, as shown on Figure 2.3: optimal methods, that yield the exact solution to the discrete optimization of Problem 1 and Problem 2, and the approximate methods, that yield an approximate solution. Described algorithms can be combined with cost functions from Section 2. Note that, depending on the chosen cost function, the computational complexity of the complete algorithm changes. As a consequence, in the following, complexity analysis is done with the assumption that applying the cost function on a sub-signal requires $\mathcal{O}(1)$ operations. Also, the practical implementations of the most important algorithms are given in pseudo-code.

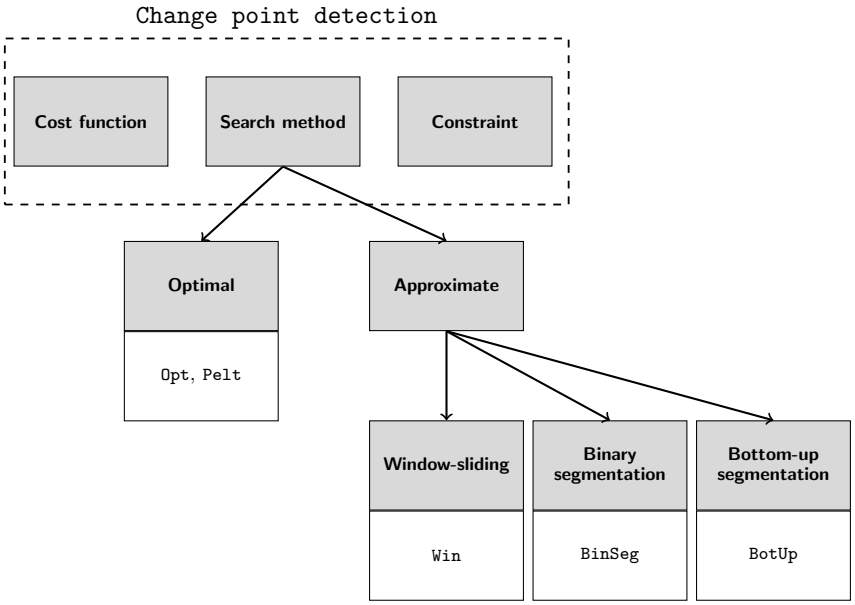


Figure 2.3: Typology of the search methods described in Section 3.

3.1 Optimal detection

Optimal detection methods find the exact solutions of Problem 1 and Problem 2. A naive approach consists in enumerating all possible segmentations of a signal, and returning the one that minimizes the objective function. However, for Problem 1, minimization is carried out over the set $\{\mathcal{T} \text{ s.t. } |\mathcal{T}| = K\}$ (which contains $\binom{T-1}{K-1}$ elements), and for Problem 2, over the set $\{\mathcal{T} \text{ s.t. } 1 \leq |\mathcal{T}| < T\}$ (which contains $\sum_{K=1}^{T-1}\binom{T-1}{K-1}$ elements). This makes exhaustive enumeration impractical, in both situations. We describe in this section two major approaches to efficiently find the exact solutions of Problem 1 and Problem 2.

3.1.1 Solution to Problem 1: `Opt`

In Problem 1, the number of change points to detect is fixed to a certain $K \geq 1$. The optimal solution to this problem can be computed efficiently, thanks to a method based on dynamic programming. The algorithm, denoted `Opt`, relies on the additive nature of the objective function $V(\cdot)$ to recursively solve sub-problems. Precisely, `Opt` is based on the following observation:

$$\begin{aligned}
\min_{|\mathcal{T}|=K} V(\mathcal{T}, y = y_{0..T}) &= \min_{0=t_0<t_1<\cdots<t_K<t_{K+1}=T} \sum_{k=0}^{K} c(y_{t_k..t_{k+1}}) \\
&= \min_{t \leq T-K} \left[c(y_{0..t}) + \min_{t=t_0<t_1<\cdots<t_{K-1}<t_K=T} \sum_{k=0}^{K-1} c(y_{t_k..t_{k+1}}) \right] \\
&= \min_{t \leq T-K} \left[c(y_{0..t}) + \min_{|\mathcal{T}|=K-1} V(\mathcal{T}, y_{t..T}) \right]
\end{aligned} \tag{2.18}$$

Intuitively, Equation 2.18 means that the first change point of the optimal segmentation is easily computed if the optimal partitions with $K-1$ elements of all sub-signals $y_{t..T}$

are known. The complete segmentation is then computed by recursively applying this observation. This strategy, described in detail in Algorithm 2.1, has a complexity of the order of $\mathcal{O}(KT^2)$ [20, 92]. Historically, `Opt` was introduced for a non-related problem [30] and later applied to change point detection, in many different contexts, such as EEG recordings [106, 108], DNA sequences [39, 139], tree growth monitoring [71], financial time-series [107, 132], radar waveforms [86], etc.

Algorithm 2.1 Algorithm `Opt`

Input: signal $\{y_t\}_{t=1}^{T}$, cost function $c(\cdot)$, number of regimes $K \geq 2$.
for all (u,v), $1 \leq u < v \leq T$ **do**
  Initialize $C_1(u,v) \leftarrow c(\{y_t\}_{t=u}^{v})$.
end for
for $k = 2,\dots,K-1$ **do**
  for all $u,v \in \{1,\dots,T\}, v-u \geq k$ **do**
    $C_k(u,v) \leftarrow \min_{u+k-1 \leq t < v} C_{k-1}(u,t) + C_1(t+1,v)$
  end for
end for
Initialize L, a list with K elements.
Initialize the last element: $L[K] \leftarrow T$.
Initialize $k \leftarrow K$.
while $k > 1$ **do**
  $s \leftarrow L(k)$
  $t^\star \leftarrow \arg\min_{k-1 \leq t < s} C_{k-1}(1,t) + C_1(t+1,s)$
  $L(k-1) \leftarrow t^\star$
  $k \leftarrow k-1$
end while
Remove T from L
Output: set L of estimated breakpoint indexes.

Related search methods. Several extensions of `Opt` have been proposed in the literature. The proposed methods still find the exact solution to Problem 1.

- The first extension is the "forward dynamic programming" algorithm [71]. Contrary to `Opt`, which returns a single partition, the "forward dynamic programming" algorithm computes the top L ($L \geq 1$) most probable partitions (i.e. with lowest sum of costs). The resulting computational complexity is $\mathcal{O}(LKT^2)$ where L is the number of computed partitions. This method is designed as a diagnostic tool: change points present in many of the top partitions are considered very likely, while change points present in only a few of the top partitions might not be as relevant. Thanks to "forward dynamic programming", insignificant change points are trimmed and overestimation of the number of change point is corrected [71], at the expense of a higher computational burden. It is applied on tree growth monitoring time series [71] that are relatively short with around a hundred samples.

- The "pruned optimal dynamic programming" procedure [139] is an extension of `Opt` that relies on a pruning rule to discard indexes that can never be change points.

Thanks to this trick, the set of potential change point indexes is reduced. All described cost functions can be plugged into this method. As a result, longer signals can be handled, for instance long array-based DNA copy number data (up to 10^6 samples, with the quadratic error cost function) [139]. However, worst case complexity remains of the order of $\mathcal{O}(KT^2)$.

3.1.2 Solution to Problem 2: `Pelt`

In Problem 2, the number of changes point is unknown, and the objective function to minimize is the penalized sum of costs. A naive approach consists in applying `Opt` for $K = 1, \dots, K_{\max}$ for a sufficiently large $K_{\max}$, then choosing among the computed segmentations the one that minimizes the penalized problem. This would prove computational cumbersome because of the quadratic complexity of the resolution method `Opt`. Fortunately a faster method exists for a general class of penalty functions, namely linear penalties. Formally, linear penalties are linear functions of the number of change points, meaning that

$$\text{pen}(\mathcal{T}) = \beta|\mathcal{T}| \tag{2.19}$$

where $\beta > 0$ is a smoothing parameter. (More details on such penalties can be found in Section 4.1.) The algorithm `Pelt` (for "Pruned Exact Linear Time") [98] was introduced to find the exact solution of Problem 2, when the penalty is linear (2.19). This approach considers each sample sequentially and, thanks to an explicit pruning rule, may or may not discard it from the set of potential change points. Precisely, for two indexes t and s ($t < s < T$), the pruning rule is given by:

$$\text{if} \quad \left[\min_{\mathcal{T}} V(\mathcal{T}, y_{0..t}) + \beta|\mathcal{T}|\right] + c(y_{t..s}) \geq \left[\min_{\mathcal{T}} V(\mathcal{T}, y_{0..s}) + \beta|\mathcal{T}|\right] \quad \text{holds,}$$
$$\text{then } t \text{ cannot be the last change point prior to } T. \tag{2.20}$$

This results in a considerable speed-up: under the assumption that regime lengths are randomly drawn from a uniform distribution, the complexity of `Pelt` is of the order $\mathcal{O}(T)$. The detailed algorithm can be found in Algorithm 2.2. An extension of `Pelt` is described in [77] to solve the linearly penalized change point detection for a range of smoothing parameter values $[\beta_{\min}, \beta_{\max}]$. `Pelt` has been applied on DNA sequences [83, 121], physiological signals [78], and oceanographic data [98].

Algorithm 2.2 Algorithm `Pelt`

Input: signal $\{y_t\}_{t=1}^T$, cost function $c(\cdot)$, penalty value β.
Initialize Z a $(T+1)$-long array; $Z[0] \leftarrow -\beta$.
Initialize $L[0] \leftarrow \emptyset$.
Initialize $\chi \leftarrow \{0\}$. ▷ Admissible indexes.
for $t = 1, \dots, T$ **do**
  $\hat{t} \leftarrow \arg\min_{s \in \chi} \left[Z[s] + c(y_{s..t}) + \beta\right]$.
  $Z[t] \leftarrow \left[Z[\hat{t}] + c(y_{\hat{t}..t}) + \beta\right]$
  $L[t] \leftarrow L[\hat{t}] \cup \{\hat{t}\}$.
  $\chi \leftarrow \{s \in \chi \,:\, Z[s] + c(y_{s..t}) \leq Z[t]\} \cup \{t\}$
end for
Output: set $L[T]$ of estimated breakpoint indexes.

3.2 Approximate detection

When the computational complexity of optimal methods is too great for the application at hand, one can resort to approximate methods. In this section, we describe three major types of approximate segmentation algorithms, namely window-based methods, binary segmentation and bottom-up segmentation. All described procedures fall into the category of sequential detection approaches, meaning that they return a single change point estimate $\hat{t}^{(k)}$ ($1 \leq \hat{t}^{(k)} < T$) at the k-th iteration. (In the following, the subscript $\cdot^{(k)}$ refers to the k-th iteration of a sequential algorithm.) Such methods can be used to solve (approximately) either Problem 1 or Problem 2. Indeed, if the number $K^\star$ of changes is known, $K^\star$ iterations of a sequential algorithm are enough to retrieve a segmentation with the correct number of changes. If $K^\star$ is unknown, the sequential algorithm is run until an appropriate stopping criterion is met.

3.2.1 Window sliding

The window-sliding algorithm, denoted `Win`, is a fast approximate alternative to optimal methods. It consists in computing the discrepancy between two adjacent windows that slide along the signal y. For a given cost function $c(\cdot)$, this discrepancy between two sub-signals is given by

$$d(y_{a..t}, y_{t..b}) = c(y_{a..b}) - c(y_{a..t}) - c(y_{t..b}) \quad (1 \leq a < t < b \leq T). \tag{2.21}$$

When the two windows cover dissimilar segments, the discrepancy reaches large values, resulting in a peak. In other other words, for each index t, `Win` measures the discrepancy between the immediate past ("left window") and the immediate future ("right window"). Once the complete discrepancy curve has been computed, a peak search procedure is performed to find change point indexes. The complete `Win` algorithm is given in Algorithm 2.3 and a schematic view is displayed on Figure 2.4. The main benefits of `Win` are its low complexity (linear in the number of samples) and ease of implementation.

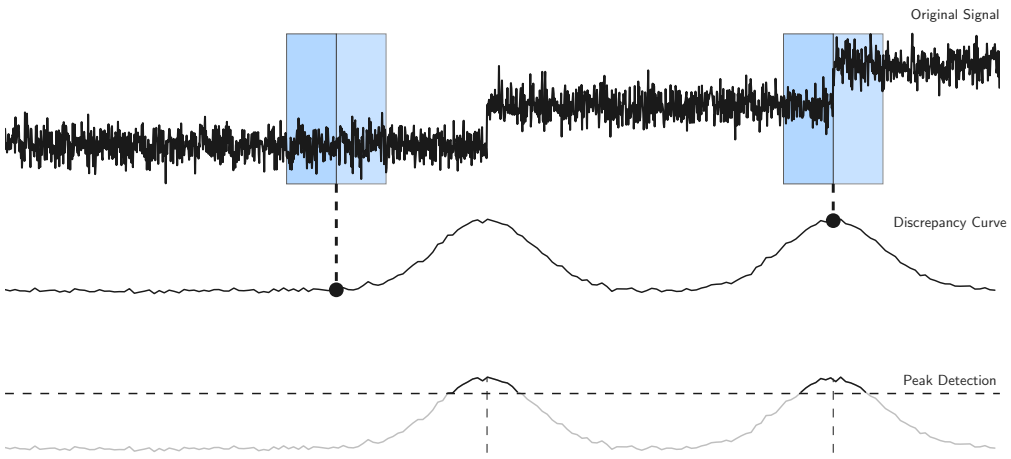


Figure 2.4: Schematic view of `Win`

Algorithm 2.3 Algorithm `Win`

Input: signal $\{y_t\}_{t=1}^T$, cost function $c(\cdot)$, half-window width w, peak search procedure `PKSearch`.
Initialize $Z \leftarrow [0, 0, \dots]$ a T-long array filled with 0. ▷ Score list.
for $t = w, \dots, T - w$ **do**
  $p \leftarrow (t - w)..t$.
  $q \leftarrow t..(t + w)$.
  $r \leftarrow (t - w)..(t + w)$.
  $Z[t] \leftarrow c(y_r) - [c(y_p) + c(y_q)]$.
end for
$L \leftarrow$ `PKSearch`(Z) ▷ Peak search procedure.
Output: set L of estimated breakpoint indexes.

In the literature, the discrepancy measure $d(\cdot, \cdot)$ is often derived from a two-sample statistical test (see Remark 2.5), and not from a cost function, as in (2.21). However, the two standpoints are generally equivalent: for instance, using c_{L_2}, $c_{\text{i.i.d.}}$ or c_{kernel} is respectively equivalent to applying a Student t-test [29], a generalized likelihood ratio (GLR) [44] test and a kernel Maximum Mean Discrepancy (MMD) test [70]. As a consequence, practitioners can capitalize on the vast body of work in the field of statistical tests to obtain asymptotic distributions for the discrepancy measure [42, 70, 114, 117], and sensible calibration strategies for important parameters of `Win` (such as the window size or the peak search procedure). `Win` has been applied in numerous contexts: for instance, on biological signals [37, 61, 75, 91, 162], on network data [114, 117], on speech time series [2, 57, 75] and on financial time series [29, 46, 95]. It should be noted that certain window-based detection methods in the literature rely on a discrepancy measure which is not related to a cost function, as in (2.21) [74, 75, 96, 115]. As a result, those methods, initially introduced in the online detection setting, cannot be extended to work with optimal algorithms (`Opt`, `Pelt`).

Remark 2.5 (Two-sample test)**.** *A two-sample test (or homogeneity test) is a statistical hypothesis testing procedure designed to assess whether two populations of samples are identical in distribution. Formally, consider two sets of i.i.d. $\mathbb{R}^d$-valued random samples $\{x_t\}_t$ and $\{z_t\}_t$. Denote by $\mathbb{P}_x$ the distribution function of the x_t and by $\mathbb{P}_z$, the distribution function of the z_t. A two-sample test procedure compares the two following hypotheses:*

$$\begin{aligned} H_0 &: \quad \mathbb{P}_x = \mathbb{P}_z \\ H_1 &: \quad \mathbb{P}_x \neq \mathbb{P}_z. \end{aligned} \tag{2.22}$$

A general approach is to consider a probability (pseudo)-metric $d(\cdot, \cdot)$ on the space of probability distributions on $\mathbb{R}^d$. Well-known examples of such a metric include the Kullback-Leibler divergence, the Kolmogorov-Smirnov distance, the Maximum Mean Discrepancy (MMD), etc. Observe that, under the null hypothesis, $d(\mathbb{P}_x, \mathbb{P}_z) = 0$. The testing procedure consists in computing the empirical estimates $\widehat{\mathbb{P}}_x$ and $\widehat{\mathbb{P}}_z$ and rejecting H_0 for "large" values of the statistics $d(\widehat{\mathbb{P}}_x, \widehat{\mathbb{P}}_z)$. This general formulation relies on a consistent estimation of arbitrary distributions from a finite number of samples. In the parametric setting, additional assumptions are made on the distribution functions: for instance, Gaussian assumption [29, 42, 43], exponential family assumption [63, 136], etc. In the non-parametric setting, the distributions are only

assumed to be continuous. They are not directly estimated; instead, the statistics $d(\widehat{\mathbb{P}}_x, \widehat{\mathbb{P}}_z)$ *are computed [50, 70, 75, 115].*
In the context of single *change point detection, the two-sample test setting is adapted to assess whether a distribution change has occurred at some instant in the input signal. Practically, for a given index* t*, the homogeneity test is performed on the two populations* $\{y_s\}_{s\leq t}$ *and* $\{y_s\}_{s>t}$*. The estimated change point location is given by*

$$\hat{t} = \arg\max_t d(\widehat{\mathbb{P}}_{\bullet\leq t}, \widehat{\mathbb{P}}_{\bullet>t}) \tag{2.23}$$

where $\widehat{\mathbb{P}}_{\bullet\leq t}$ *and* $\widehat{\mathbb{P}}_{\bullet>t}$ *are the empirical distributions of respectively* $\{y_s\}_{s\leq t}$ *and* $\{y_s\}_{s>t}$*.*

3.2.2 Binary segmentation

Binary segmentation, denoted `BinSeg`, is a well-known alternative to optimal methods [146], because it is conceptually simple and easy to implement [43, 98, 126]. `BinSeg` is a greedy sequential algorithm, outlined as follows. The first change point estimate $\hat{t}^{(1)}$ is given by

$$\hat{t}^{(1)} := \arg\min_{1\leq t<T-1} \underbrace{c(y_{0..t}) + c(y_{t..T})}_{V(\mathcal{T}=\{t\})}. \tag{2.24}$$

This operation is "greedy", in the sense that it searches the change point that lowers the most the sum of costs. The signal is then split in two at the position of $\hat{t}^{(1)}$; the same operation is repeated on the resulting sub-signals until a stopping criterion is met. A schematic view of the algorithm is displayed on Figure 2.5 and an implementation is given in Algorithm 2.4. The complexity of `BinSeg` is of the order of $\mathcal{O}(T\log T)$. This low complexity comes at the expense of optimality: in general, `BinSeg`'s output is only an approximation of the optimal solution. As argued in [11, 98], the issue is that the estimated change points $\hat{t}^{(k)}$ are not estimated from homogeneous segments and each estimate depends on the previous ones. Change points that are close are imprecisely detected especially [90]. Applications of `BinSeg` range from financial time series [11, 42, 43, 67, 111] to context recognition for mobile devices [81] and array-based DNA copy number data [126, 134].

Related search methods. Several extensions of `BinSeg` have been proposed to improve detection accuracy.

- Circular binary segmentation [126] is a well-known extension of `BinSeg`. This method is also a sequential detection algorithm that splits the original at each step. Instead of searching for a single change point in each sub-signal, circular binary segmentation searches two change points. Within each treated sub-segment, it assumes a so-called "epidemic change model": the parameter of interest shifts from one value to another at the first change point and returns to the original value at the second change point. The algorithm is dubbed "circular" because, under this model, the sub-segment has its two ends (figuratively) joining to form a circle. Practically, this method has been combined with c_{L_2} C2, to detect changes in the mean of array-based DNA copy number data [104, 126, 164]. A faster version of the original algorithm is described in [159].

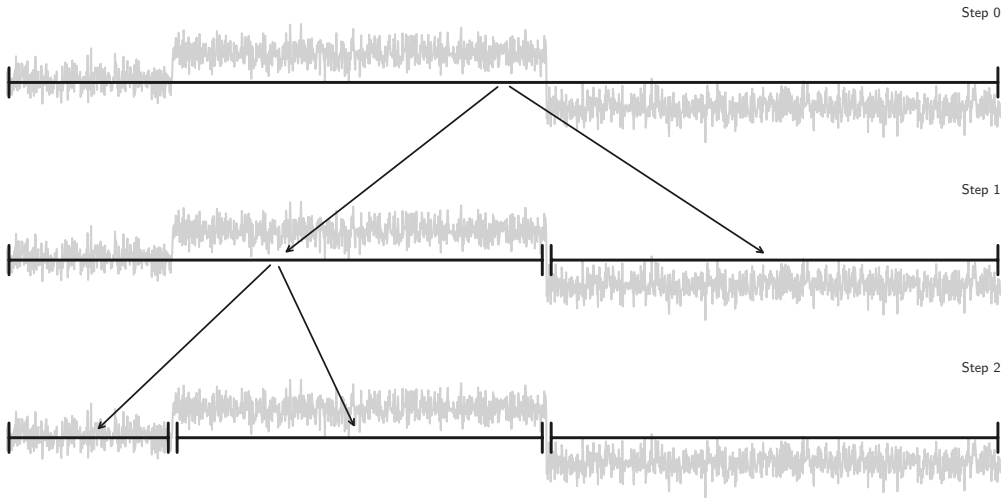


Figure 2.5: Schematic example of `BinSeg`

Algorithm 2.4 Algorithm `BinSeg`

Input: signal $\{y_t\}_{t=1}^T$, cost function $c(\cdot)$, stopping criterion.
Initialize $L \leftarrow \{\ \}$. ▷ Estimated breakpoints.
repeat
  $k \leftarrow |L|$. ▷ Number of breakpoints
  $t_0 \leftarrow 0$ and $t_{k+1} \leftarrow T$ ▷ Dummy variables.
  if $k > 0$ **then**
    Denote by t_i $(i = 1,\dots,k)$ the elements (in ascending order) of L, i.e. $L = \{t_1,\dots,t_k\}$.
  end if
  Initialize G a $(k+1)$-long array. ▷ list of gains
  for $i = 0,\dots,k$ **do**
    $G[i] \leftarrow c(y_{t_i..t_{i+1}}) - \min_{t_i<t<t_{i+1}} [c(y_{t_i..t}) + c(y_{t..t_{i+1}})]$.
  end for
  $\widehat{i} \leftarrow \arg\max_i G[i]$
  $\hat{t} \leftarrow \arg\min_{t_{\widehat{i}}<t<t_{\widehat{i}+1}} [c(y_{t_{\widehat{i}}..t}) + c(y_{t..t_{\widehat{i}+1}})]$.
  $L \leftarrow L \cup \{\hat{t}\}$
until stopping criterion is met.
Output: set L of estimated breakpoint indexes.

- Another extension of `BinSeg` is the wild binary segmentation algorithm [67]. In a nutshell, a single point detection is performed on multiple intervals with start and end points that are drawn uniformly. Small segments are likely to contain at most one change but have lower statistical power, while the opposite is true for long segments. After a proper weighting of the change score to account for the differences on sub-signals' length, the algorithm returns the most "pronounced" ones, i.e. those that lower the most the sum of costs. An important parameter of this method is

the number of random sub-segments to draw. Wild binary search is combined with c_{L_2} C2 to detect mean-shifts of univariate piecewise constant signals (up to 2000 samples) [67].

3.2.3 Bottom-up segmentation

Bottom-up segmentation, denoted `BotUp`, is the natural counterpart of `BinSeg`. Contrary to `BinSeg`, `BotUp` starts by splitting the original signal in many small sub-signals and sequentially merges them until there remain only K change points. At every step, all potential change points (indexes separating adjacent sub-segments) are ranked by the discrepancy measure $d(\cdot, \cdot)$, defined in 2.21, between the segments they separate. Change points with the lowest discrepancy are then deleted, meaning that the segments they separate are merged. `BotUp` is often dubbed a "generous" method, by opposition to `BinSeg`, which is "greedy" [94]. A schematic view of the algorithm is displayed on Figure 2.6 and an implementation is provided in Algorithm 2.5. Its benefits are its linear computational complexity and conceptual simplicity. However, if a true change point does not belong to the original set of indexes, `BotUp` never considers it. Moreover, in the first iterations, the merging procedure can be unstable because it is performed on small segments, for which statistical significance is smaller. In the literature, `BotUp` is somewhat less studied than its counterpart, `BinSeg`: no theoretical convergence study is available. It has been applied on speech time series to detect mean and scale shifts [46]. Besides, the authors of [94] have found that `BotUp` outperforms `BinSeg` on ten different data sets such as physiological signals (ECG), financial time-series (exchange rate), industrial monitoring (water levels), etc.

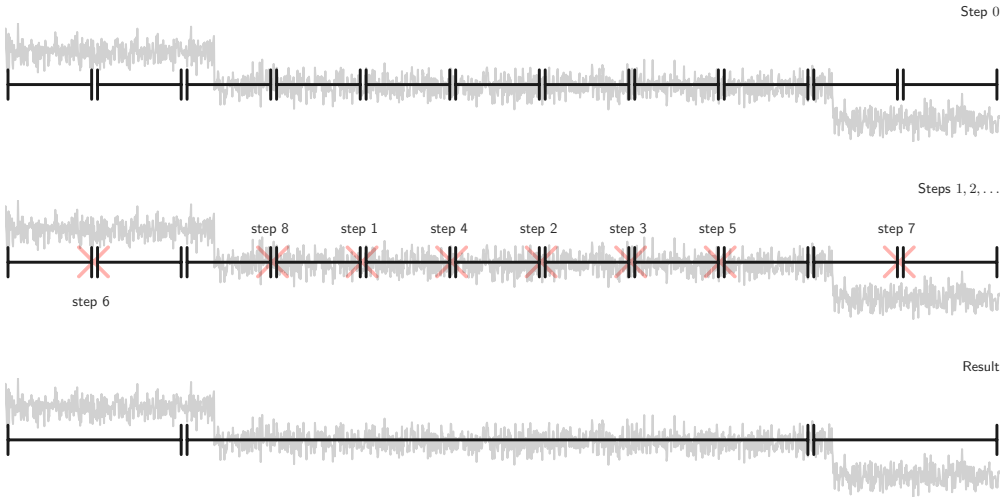


Figure 2.6: Schematic view of `BotUp`

Algorithm 2.5 Algorithm `BotUp`

Input: signal $\{y_t\}_{t=1}^T$, cost function $c(\cdot)$, stopping criterion, grid size $\delta > 2$.
Initialize $L \leftarrow \{\delta, 2\delta, \dots, (\lfloor T/\delta \rfloor - 1)\,\delta\}$. ▷ Estimated breakpoints.
repeat
  $k \leftarrow |L|$. ▷ Number of breakpoints
  $t_0 \leftarrow 0$ and $t_{k+1} \leftarrow T$ ▷ Dummy variables.
  Denote by t_i $(i = 1, \dots, k)$ the elements (in ascending order) of L, i.e. $L = \{t_1, \dots, t_k\}$.
  Initialize G a $(k-1)$-long array. ▷ list of gains
  for $i = 1, \dots, k-1$ **do**
    $G[i-1] \leftarrow c(y_{t_{i-1}..t_{i+1}}) - [c(y_{t_{i-1}..t_i}) + c(y_{t_i..t_{i+1}})]$.
  end for
  $\widehat{i} \leftarrow \arg\min_i G[i]$
  Remove $t_{\widehat{i}+1}$ from L.
until stopping criterion is met.
Output: set L of estimated breakpoint indexes.

4 Estimating the number of changes

This section presents the third defining element of change detection methods, namely the constraint on the number of change points. Here, the number of change points is assumed to be unknown (Problem 2). Existing procedures are organized by the penalty function that they are based on. Common heuristics are also described. The organization of this section is schematically shown in Figure 2.7.

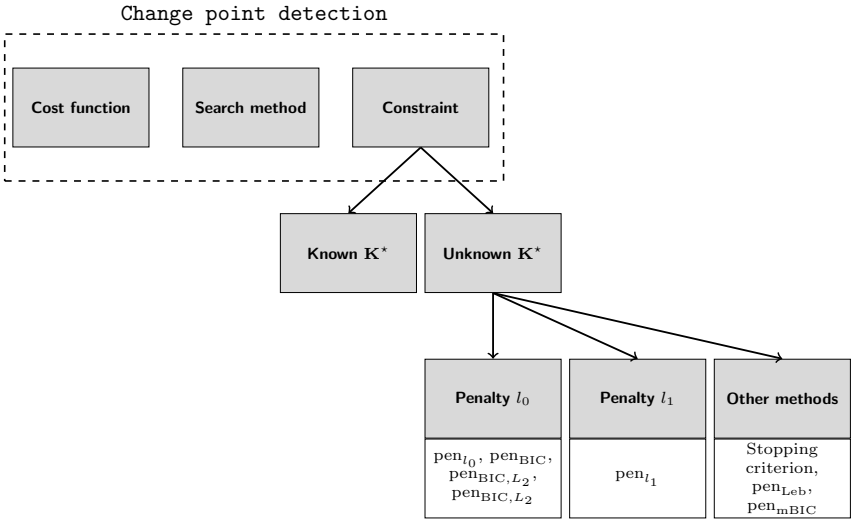


Figure 2.7: Typology of the constraints (on the number of change points) described in Section 4.

4.1 Linear penalty

Arguably the most popular choice of penalty [98], the linear penalty (also known as l_0 penalty) generalizes several well-known criteria from the literature such as the Bayesian

Information Criterion (BIC) and the Akaike Information Criterion (AIC) [166, 167]. The linear penalty, denoted pen_{l_0}, is formally defined as follows.

Definition 2.15 (pen_{l_0})**.** *The penalty function pen_{l_0} is given by*

$$pen_{l_0}(\mathcal{T}) := \beta|\mathcal{T}| \tag{2.25}$$

where $\beta > 0$ is the smoothing parameter.

Intuitively, the smoothing parameter controls the trade-off between complexity and goodness-of-fit (measured by the sum of costs): low values of β favour segmentations with many regimes and high values of β discard most change points.

Calibration. From a practical standpoint, once the cost function has been chosen, the only parameter to calibrate is the smoothing parameter. Several approaches, based on model selection, can be found in the literature: they assume a model on the data, for instance (M1), (M2), (M3), and choose a value of β that optimizes a certain statistical criterion. The best-known example of such an approach is BIC, which aims at maximizing the constrained log-likelihood of the model. The exact formulas of several linear penalties, derived from model selection procedures, are given the following paragraph. Conversely, when no model is assumed, different heuristics are applied to tune the smoothing parameter. For instance, one can use a procedure based on cross-validation [4] or the slope heuristics [33]. In [82], an original supervised algorithm is proposed: the chosen β is the one that minimizes an approximation of the segmentation error on an annotated set of signals.

Related penalties. A number of model selection criteria are special cases of the linear penalty pen_{l_0}. For instance, under Model (M1) (i.i.d. with piecewise constant distribution), the constrained likelihood that is derived from the BIC and the penalized sum of costs are formally equivalent, upon setting $c = c_{\text{i.i.d.}}$ and $\text{pen} = \text{pen}_{\text{BIC}}$, where pen_{BIC} is defined as follows.

Definition 2.16. *The penalty function pen_{BIC} is given by*

$$pen_{BIC}(\mathcal{T}) := \frac{p}{2}\log T\ |\mathcal{T}| \tag{2.26}$$

where $p \geq 1$ is the dimension of the parameter space in (M1).

In the extensively studied model of an univariate Gaussian signal, with fixed variance σ^2 and piecewise constant mean, the penalty pen_{BIC} becomes pen_{L_2}, defined below. Historically, it was one of the first penalties introduced for change point detection [144, 166].

Definition 2.17. *The penalty function pen_{BIC,L_2} is given by*

$$pen_{BIC,L_2}(\mathcal{T}) := \sigma^2 \log T\ |\mathcal{T}|. \tag{2.27}$$

where σ is the standard deviation and T is the number of samples.

In the same setting, AIC, which is a generalization of Mallows' C_p [122], also yields a linear penalty, namely $\text{pen}_{\text{AIC},L_2}$, defined as follows.

Definition 2.18. *The penalty function pen_{AIC,L_2} is given by*

$$pen_{AIC,L_2}(\mathcal{T}) := \sigma^2\ |\mathcal{T}|. \tag{2.28}$$

where σ is the standard deviation.

4.2 Fused lasso

For the special case where the cost function is c_{L_2}, a faster alternative to pen_{l_0} can be used. To that end, the l_0 penalty is relaxed to a l_1 penalty [73, 160]. The resulting penalty function, denoted pen_{l_1}, is defined as follows.

Definition 2.19 (pen_{l_1}). *The penalty function pen_{l_1} is given by*

$$pen_{l_1}(\mathcal{T}) := \beta \sum_{k=1}^{|\mathcal{T}|} \left\| \bar{y}_{t_{k-1}..t_k} - \bar{y}_{t_k..t_{k+1}} \right\|_1 \tag{2.29}$$

where $\beta > 0$ is the smoothing parameter, the t_k are the elements of $\mathcal{T}$ and $\bar{y}_{t_{k-1}..t_k}$ is the empirical mean of sub-signal $y_{t_{k-1}..t_k}$.

This relaxation strategy (from l_0 to l_1) is shared with many developments in machine learning, for instance sparse regression, compressive sensing, sparse PCA, dictionary learning [76], where pen_{l_1} is also referred to as the fused lasso penalty. In numerical analysis and image denoising, it is also known as the total variation regularizer [73, 145, 160]. Thanks to this relaxation, the optimization of the penalized sum of costs (2.3) in Problem 2 is transformed into a convex optimization problem, which can be solved efficiently using Lars (for "least absolute shrinkage and selection operator") [73, 160]. The resulting complexity is of this order of $\mathcal{O}(T \log T)$ [76, 151]. From a theoretical standpoint, under the mean-shift model (piecewise constant signal with Gaussian white noise), the estimated change point fractions are asymptotically consistent [73]. This result is demonstrated for an appropriately converging sequence of values of β. This consistency property is obtained even though classical assumptions from the Lasso regression framework (such as the irrepresentable condition) are not satisfied [73]. In the literature, pen_{l_1}, combined with c_{L_2}, is applied on DNA sequences [83, 160] and speech signals [3].

4.3 Complex penalties

Several other penalty functions can be found in the literature. However they are more complex, in the sense that the optimization of the penalized sum of cost is not tractable. In practice, the solution is found by computing the optimal segmentations with K change points, with $K = 1, 2, \ldots, K_{\max}$ for a sufficiently large $K_{\max}$, and returning the one that minimizes the penalized sum of costs. When possible, the penalty can also be approximated by a linear penalty, in which case, `Pelt` can be used. In this section, we describe two examples of complex penalties. Both originate from theoretical considerations, under the univariate mean-shift model, with the cost function c_{L_2}. The first example is the modified BIC criterion (mBIC) [170], which consists in maximizing the asymptotic posterior probability of the data. The resulting penalty function, denoted pen_{mBIC}, depends on the number and repartition of the change point indexes: intuitively, it favours evenly spaced change points.

Definition 2.20 (pen_{mBIC}). *The penalty function pen_{mBIC} is given by*

$$pen_{mBIC}(\mathcal{T}) := 3|\mathcal{T}| \log T + \sum_{k=0}^{|\mathcal{T}|+1} \log\left(\frac{t_{k+1} - t_k}{T}\right) \tag{2.30}$$

where the t_k are the elements of $\mathcal{T}$.

In [112], a model selection procedure leads to another complex penalty function, namely pen_{Leb}. Upon using this penalty function, the penalized sum of costs satisfied a so-called oracle inequality, which holds in a non-asymptotic setting, contrary to the other penalties previously described.

Definition 2.21 (pen_{Leb}). *The cost function pen_{Leb} is given by*

$$pen_{Leb}(\mathcal{T}) := \frac{|\mathcal{T}|+1}{T}\sigma^2(a_1 \log \frac{|\mathcal{T}|+1}{T} + a_2) \tag{2.31}$$

where $a_1 > 0$ and $a_2 > 0$ are positive parameters and σ^2 is the noise variance.

5 Summary table

This chapter of literature review is summarized in Table 2.2. When applicable, each publication is associated with a search method (such as `Opt`, `Pelt`, `BinSeg` or `Win`); this is a rough categorization rather than an exact implementation. Note that `Pelt` (introduced in 2012) is sometimes associated with publications prior to 2012. It is because some linear penalties [122, 170] were introduced long before `Pelt` was, and authors then resorted to quadratic (at best) algorithms. Nowadays, the same results can be obtained faster with `Pelt`. A guide of computational complexity is also provided. Quadratic methods are the slowest and have only one star while linear methods are given three stars. Algorithms for which the number of change points is an explicit input parameter work under the "known K" assumption. Algorithms that can be used even if the number of change points is unknown work under the "unknown K" assumption. (Certain methods can accommodate both situations.) Some methods are implemented and available online. We refer the reader to Table 9.1 in Chapter 9 for a detailed summary of available libraries.

6 Conclusion

In this chapter, we have reviewed numerous methods to perform change point detection, organized within a common framework. Precisely, all methods are described as a collection of three elements: a cost function, a search method and a constraint on the number of changes to detect. This approach is intended to facilitate prototyping of change point detection methods: for a given segmentation task, one can pick among the described elements to design an algorithm that fits its use-case. Most detection procedures described above are available within the Python language from the package `ruptures` (ctruong.perso.math.cnrs.fr/ruptures). Additional information can be found in Chapter 9, which is dedicated to the description of this package.

Publication	Search method	Cost function	Known K		Scalability (w.r.t. T)	Package	Additional information
			Yes	No			
Sen and Srivastava (1975), Vostrikova (1981)	`BinSeg`	c_{L_2}	✓	-	★★★	✓	
Yao (1988)	`Opt`	c_{L_2}	-	✓	★☆☆	-	Bayesian information criterion (BIC)
Basseville and Nikiforov (1993)	`Opt`	$c_{\text{i.i.d.}}, c_{L_2}$	-	-	★★★	-	single change point
Bai (1994), Bai and Perron (2003)	`Opt`	c_{linear,L_2}	-	-	★★☆	-	single change point
Bai (1995)	`Opt`	c_{linear,L_1}	-	-	★★☆	-	single change point
Lavielle (1998)	`Opt`	c_{AR}	✓	-	★☆☆	-	
Bai (2000)	`Opt`	c_{AR}	✓	-	★☆☆	-	
Birgé and Massart (2001), Birgé and Massart (2007)	`Opt`	c_{L_2}	-	✓	★☆☆	-	model selection
Bai and Perron (2003)	`Opt`	c_{L_2}	✓	-	★☆☆	-	
Olshen et al. (2004), Venkatraman and Olshen (2007)	`BinSeg`	c_{L_2}	✓	✓	★★★	✓	
Lebarbier (2005)	`Opt`	c_{L_2}	-	✓	★☆☆	-	model selection
Desobry et al. (2005)	`Win`	c_{kernel}	-	✓	★★★	-	dissimilarity measure (one-class SVM), see Remark 2.5
Harchaoui and Cappé (2007)	`Opt`	c_{kernel}, c_{rbf}	✓	-	★☆☆	-	
Zhang and Siegmund (2007)	`Pelt`	c_{L_2}	-	✓	★★☆	-	modified BIC
Harchaoui et al. (2009)	`Win`	-	✓	✓	★★★	-	dissimilarity measure (Fisher discriminant), see Remark 2.5
Lévy-Leduc and Roueff (2009), Lung-Yut-Fong et al. (2012)	`Win`	c_{rank}	✓	✓	★★★	✓	dissimilarity measure (rank-based), see Remark 2.5
Bai (2010)	`Opt`	c_{L_2}, c_{Σ}	-	-	★★☆	-	single change point
Vert and Bleakley (2010)	Fused Lasso	c_{L_2}	-	✓	★★★	-	Tikhonov regularization
Harchaoui and Lévy-Leduc (2010)	Fused Lasso	c_{L_2}	-	✓	★★★	-	total variation regression (pen_{l_1})
Arlot et al. (2012)	`Opt`	c_{kernel}, c_{rbf}	✓	✓	★☆☆	-	
Killick et al. (2012)	`Pelt`	any $c(\cdot)$	-	✓	★★☆	✓	
Angelosante and Giannakis (2012)	Fused Lasso	c_{AR}	-	✓	★★★	-	Tikhonov regularization
Liu et al. (2013)	`Win`	-	-	✓	★★★	-	dissimilarity measure (density ratio), see Remark 2.5
Hocking et al. (2013)	`Pelt`	c_{L_2}	-	✓	★★☆	-	supervised method to learn a penalty level (pen_{l_0})
Fryzlewicz (2014)	`BinSeg`	c_{L_2}	✓	✓	★★★	✓	univariate signal
Lajugie et al. (2014)	`Opt`	c_M	✓	-	★☆☆	-	supervised method to learn a suitable metric
Frick et al. (2014)	`BinSeg`	$c_{\text{i.i.d.}}$	✓	✓	★★★	✓	exponential distributions family
Lung-Yut-Fong et al. (2015)	`Opt`	c_{rank}	✓	-	★☆☆	✓	
Garreau and Arlot (2017)	`Pelt`	c_{kernel}, c_{rbf}	✓	✓	★☆☆	-	
Haynes et al. (2017)	`Pelt`	any $c(\cdot)$	-	✓	★★☆	-	
Chakar et al. (2017)	`Pelt`	c_{AR}	✓	✓	★☆☆	✓	

Table 2.2: Summary table of literature review.

3

Evaluation framework: metrics and data sets

Contents

Abstract

This chapter presents the evaluation framework that is used throughout this manuscript to experimentally compare segmentation algorithms. The shape of this framework is largely motivated by the challenges met by Cognac. In addition to several error metrics, three data sets are described: *Gait*, *MeanShift* and *FreqShift*. The *Gait* data set contains real-world signals collected by Cognac-G for the study of the human walking motion. *MeanShift* and *FreqShift* contain synthetic signals. Both can be seen as idealized approximations of *Gait*.

1 Motivations

As described in Chap. 1 : Introduction, signals collected in the context of Cognac-G are from monitored subjects who undergo a medical protocol. Typically, a clinician asks a patient to perform several consecutive physical exercises while some sensors record some physiological and bio-mechanical variables (e.g. heart rate, oxygen uptake, body acceleration). Those time series are made of consecutive phases, whose temporal boundaries must be estimated, thanks to change point detection methods. In order to quantify the evolution of the patient during the protocol, the monitoring signal is segmented, meaning that it is split in sub-signals, each corresponding to a coherent phase (for instance, a single exercise). Certain features of interest are then computed for each phase. This segmentation step is critical in the contextualization of long time series.
The objective of this thesis is to provide detection algorithms that follow the general principles that have emerged in the context of Cognac-G (see, in the introductory chapter, Sec. 3 : Change point detection for physiological data). In order to compare our contributions

to state-of-the-art methods, a crucial element is a consistent evaluation framework. We list below the important criteria by which segmentation performances are assessed, in this framework.

- Depending on the clinical protocol, the *number of change points* may be unknown, in which case it must be estimated by the segmentation algorithm. Missing or adding a change can respectively result in heterogeneous sub-signals and shorter sub-signals. Both of those situations can adversely influence any subsequent data treatment, if it relies on homogeneous regimes.
- *Accuracy* of a detection method is the quality of correctly estimating the location of change points, on average. If a set of changes points is inaccurately estimated, the associated regimes are not homogeneous, which, again, adversely influences any subsequent data treatment.
- In an effort to develop automatic procedures, a great emphasis is put on *robustness*. In order to be applied on large data sets without human supervision, algorithms must be able to operate in a wide range of settings.
- The *execution time* of algorithms must comply with the constraints of daily clinical practice. This is even more important for change point detection methods, which are only a pre-processing step, and are followed by a number of other data treatments.

2 Evaluation framework

The evaluation framework is composed of evaluation metrics and three data sets. Each metric is a measure of the error made when estimating the segmentation of a signal. As for the data sets, one contains real-world time series collected in the context of Cognac-G, and the other two are synthetic approximations of the first one.
The general process by which a segmentation method is evaluated is outlined as follows. First, the algorithm at hand is applied on a testing data set of signals $y^{(l)}$ ($l = 1, \dots, L$). The change point estimates are denoted $\widehat{\mathcal{T}}^{(l)}$. Then, the change point estimates are compared to the true segmentation, denoted $\mathcal{T}^{(l)}$, using one of the metrics described below. The mean and standard deviation of the metric values (over the data set) are reported for comparison.

2.1 Evaluation metrics

Several metrics from the literature are presented below. Each metric correspond to one of the previously listed criteria by which segmentation performances are assessed. In the following, the set of true change points is denoted by $\mathcal{T}^\star = \{t_1^\star, \dots, t_{K^\star}^\star\}$, and the set of estimated change points is denoted by $\widehat{\mathcal{T}} = \{\hat{t}_1, \dots, \hat{t}_{\widehat{K}}\}$. Note that that the cardinals of each set, $K^\star$ and $\widehat{K}$, are not necessarily equal.

2.1.1 AnnotationError

The AnnotationError is simply the difference between the predicted number of change points $|\widehat{\mathcal{T}}|$ and the true number of change points $|\mathcal{T}^\star|$:

$$\text{AnnotationError} := |\widehat{K} - K^\star|. \tag{3.1}$$

This metric can be used to discriminate detection method when the number of changes is unknown.

2.1.2 HAUSDORFF

The HAUSDORFF metric measures the robustness of detection methods [35, 73]. Formally, it is equal to the greatest temporal distance between a change point and its prediction:

$$\text{HAUSDORFF}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := \max \{ \max_{\hat{t}\in\widehat{\mathcal{T}}} \min_{t^\star\in\mathcal{T}^\star} |\hat{t} - t^\star|, \max_{t^\star\in\mathcal{T}^\star} \min_{\hat{t}\in\widehat{\mathcal{T}}} |\hat{t} - t^\star| \}.$$

It is the worst error made by the algorithm that produced $\widehat{\mathcal{T}}$ and is expressed in number of samples. If this metric is equal to zero, both breakpoint sets are equal; it is large when a change point from either $\mathcal{T}^\star$ or $\widehat{\mathcal{T}}$ is far from every change point of $\widehat{\mathcal{T}}$ or $\mathcal{T}^\star$ respectively. Over-segmentation as well as under-segmentation is penalized. An illustrative example is displayed on Figure 3.1.

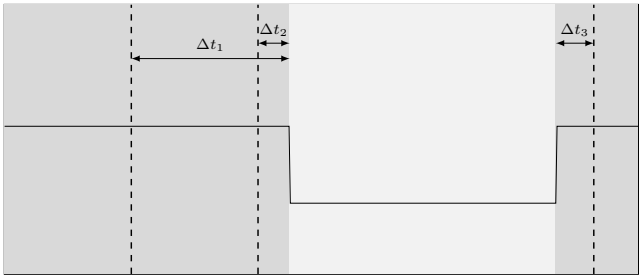


Figure 3.1: HAUSDORFF. Alternating gray areas mark the segmentation $\mathcal{T}^\star$; dashed lines mark the segmentation $\widehat{\mathcal{T}}$. Here, HAUSDORFF is equal to $\Delta t_1 = \max(\Delta t_1, \Delta t_2, \Delta t_3)$.

2.1.3 RANDINDEX

Accuracy can be measured by the RANDINDEX, which is the average similarity between the predicted breakpoint set $\widehat{\mathcal{T}}$ and the ground truth $\mathcal{T}^\star$ [105]. Intuitively, it is equal to the number of agreements between two segmentations. An agreement is a pair of indexes which are either in the same segment according to both $\widehat{\mathcal{T}}$ and $\mathcal{T}^\star$ or in different segments according to both $\widehat{\mathcal{T}}$ and $\mathcal{T}^\star$. Formally, for a breakpoint set $\mathcal{T}$, the set of grouped indexes and the set of non-grouped indexes are respectively $\text{gr}(\mathcal{T})$ and $\text{ngr}(\mathcal{T})$:

$$\text{gr}(\mathcal{T}) := \{(s,t), 1 \leq s < t \leq T \text{ s.t. } s \text{ and } t \text{ belong to the same segment according to } \mathcal{T} \},$$
$$\text{ngr}(\mathcal{T}) := \{(s,t), 1 \leq s < t \leq T \text{ s.t. } s \text{ and } t \text{ belong to different segments according to } \mathcal{T} \}.$$

The RANDINDEX is then defined as follows:

$$\text{RANDINDEX}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := \frac{|\text{gr}(\widehat{\mathcal{T}}) \cap \text{gr}(\mathcal{T}^\star)| + |\text{ngr}(\widehat{\mathcal{T}}) \cap \text{ngr}(\mathcal{T}^\star)|)}{T(T-1)}. \tag{3.2}$$

It is normalized between 0 (total disagreement) and 1 (total agreement). Originally, RANDINDEX has been introduced to evaluate clustering methods [35, 105]. An illustrative example is displayed on Figure 3.2.

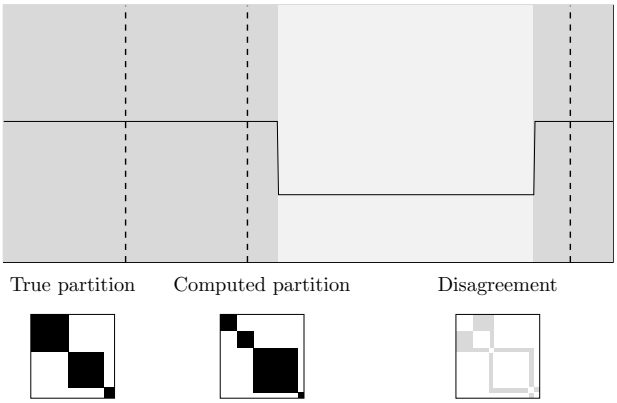


Figure 3.2: RANDINDEX. Top: alternating gray areas mark the segmentation $\mathcal{T}^\star$; dashed lines mark the segmentation $\widehat{\mathcal{T}}$. Below: representations of associated adjacency matrices and disagreement matrix. The adjacency matrix of a segmentation is the $T \times T$ binary matrix with coefficient (s,t) equal to 1 if s and t belong to the same segment, 0 otherwise. The disagreement matrix is the $T \times T$ binary matrix with coefficient (s,t) equal to 1 where the two adjacency matrices disagree, and 0 otherwise. RANDINDEX is equal to the white area (where coefficients are 0) of the disagreement matrix.

2.1.4 F1 SCORE

Another measure of accuracy is the F1 SCORE. Precision is the proportion of predicted change points that are true change points. Recall is the proportion of true change points that are well predicted. A breakpoint is considered detected up to a user-defined margin of error $M > 0$; true positives TP are true change points for which there is an estimated one at less than M samples, *i.e.*

$$\text{TP}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := \{t^\star \in \mathcal{T}^\star \mid \exists \hat{t} \in \widehat{\mathcal{T}} \text{ s.t. } |\hat{t} - t^\star| < M\}. \tag{3.3}$$

Precision PREC and recall REC are then given by

$$\text{PREC}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := |\text{TP}(\mathcal{T}^\star, \widehat{\mathcal{T}})| / \widehat{K} \quad \text{and} \quad \text{REC}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := |\text{TP}(\mathcal{T}^\star, \widehat{\mathcal{T}})| / K^\star. \tag{3.4}$$

PRECISION and RECALL are well-defined (i.e. between 0 and 1) if the margin M is smaller than the minimum spacing between two true change point indexes $t_k^\star$ and $t_{k+1}^\star$. Over-segmentation of a signal causes the precision to be close to zero and the recall close to one. Under-segmentation has the opposite effect. The F1 SCORE is the harmonic mean of precision PREC and recall REC:

$$\text{F1 SCORE}(\mathcal{T}^\star, \widehat{\mathcal{T}}) := 2 \times \frac{\text{PREC}(\mathcal{T}^\star, \widehat{\mathcal{T}}) \times \text{REC}(\mathcal{T}^\star, \widehat{\mathcal{T}})}{\text{PREC}(\mathcal{T}^\star, \widehat{\mathcal{T}}) + \text{REC}(\mathcal{T}^\star, \widehat{\mathcal{T}})}. \tag{3.5}$$

Its best value is 1 and its worse value is 0. An illustrative example is displayed on Figure 3.3.

2.2 Presentation of the data sets

The corpus is composed of a real-world data set, *Gait*, and two synthetic data sets, *MeanShift* and *FreqShift*. The two synthetic data sets provide a controlled environnement, to assess

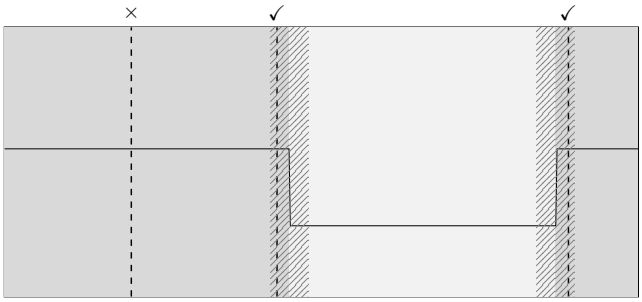


Figure 3.3: F1 SCORE. Alternating gray areas mark the segmentation $\mathcal{T}^\star$; dashed lines mark the segmentation $\widehat{\mathcal{T}}$; dashed areas mark the allowed margin of error around true change points. Here, PREC is $2/3$, REC is $2/2$ and F1 SCORE is $4/5$.

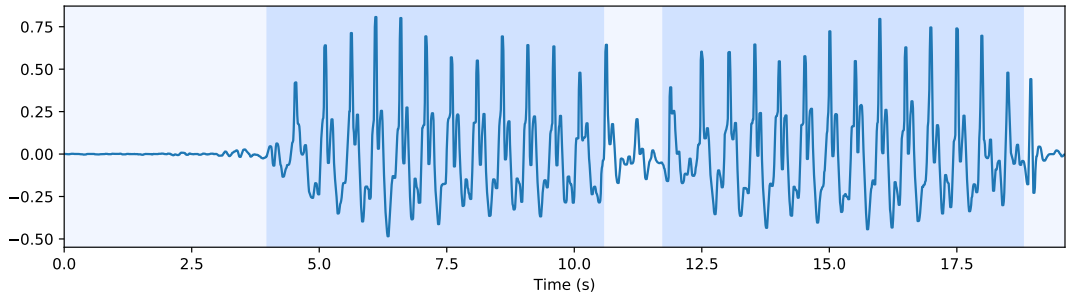
the influence of certain parameters (such as the number of samples, the noise, etc.) on the segmentation efficiency. Both *MeanShift* and *FreqShift* are designed to so as to resemble the shapes of signals from *Gait*. Each data set contains L signals $y^{(l)}$ (L depends on the data set), possibly grouped into sub-sets of equal segmentation difficulty. Each signal $y^{(l)}$ has a "true" segmentation $\mathcal{T}^{(l)}$, which is either the one used to simulate the signal for *MeanShift* and *FreqShift* or the manual segmentation provided by the medical researchers for *Gait*.

2.2.1 The *Gait* data set

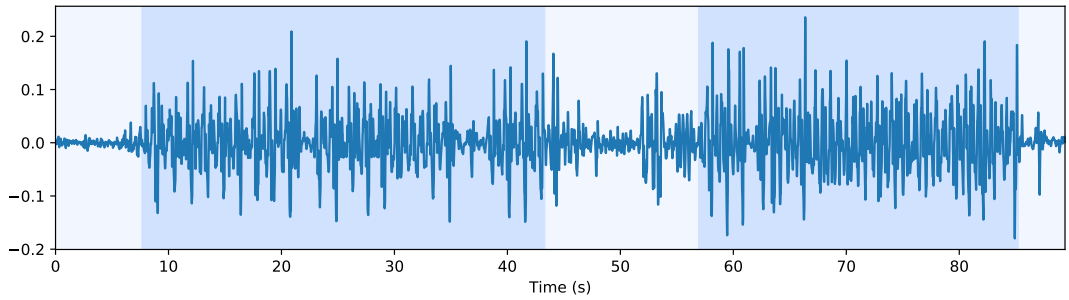
Context. Human locomotion is a complex mechanism composed of a succession of strides, steps, and phases [24, 26]. Some pathologies (such as Parkinson's disease, arthritis, stroke, obesity, diabetes, etc.) may alter the locomotion, threatening the autonomy of patients and increasing the risk of fall. Objective quantification and assessment of locomotion is therefore a crucial problem, that has been addressed in the literature by measuring the movement with several types of sensors such as inertial sensors, instrumented mat, force platforms, camera-optical tracking system or force-sensitive resistors insoles. The signals obtained from these sensors are processed (automatically or manually) to extract some features that characterize locomotion (speed, variability, smoothness, etc.). In this context, the COGNAC G team has conceived and implemented a clinical protocol for the analysis of human gait using Inertial Measurement Units (IMUs) which are composed of 3D accelerometers, 3D gyroscopes and 3D magnetometers. The main advantages of these sensors are that they are relatively low-cost, they do not require a dedicated room for the experiments, and their small size make them easy to handle in day-to-day clinical situations. The data used for the conception and testing of the method presented in this thesis has been provided by several medical departments[1]. The study was validated by a local ethic comity and both patients and control subjects gave their written consent to participate. All signals have been acquired at 100 Hz with wireless XSens MTw™ sensors located at lower back and fixed using a Velcro band designed by XSens™. All subjects were asked to (1) stand still for 6 seconds, (2) walk 10 meters at preferred walking speed on a level surface, (3) turn around, (4) walk back, (5) stand still for 2 seconds. There are 54 subjects, monitored on

[1]Service de chirurgie orthopédique et de traumatologie de l'Hôpital Européen Georges Pompidou, Assistance Publique des Hôpitaux de Paris, Service de médecine physique et de réadaptation de l'Hôpital Fernand Widal, Assistance Publique des Hôpitaux de Paris, Service de neurologie de l'Hôpital d'Instruction des Armées du Val de Grâce, Service de Santé des Armées

multiple occasions, resulting in a data set of 262 signals. Two examples of recorded signals are displayed on Figure 3.4. The two flat parts at the extremities of the signal correspond to periods when the subject is standing still. The repeated patterns represent the footsteps. Depending on the pathology (or absence of), the length of the signal varies from 20 seconds to 90 seconds.



(a) Healthy control subject.

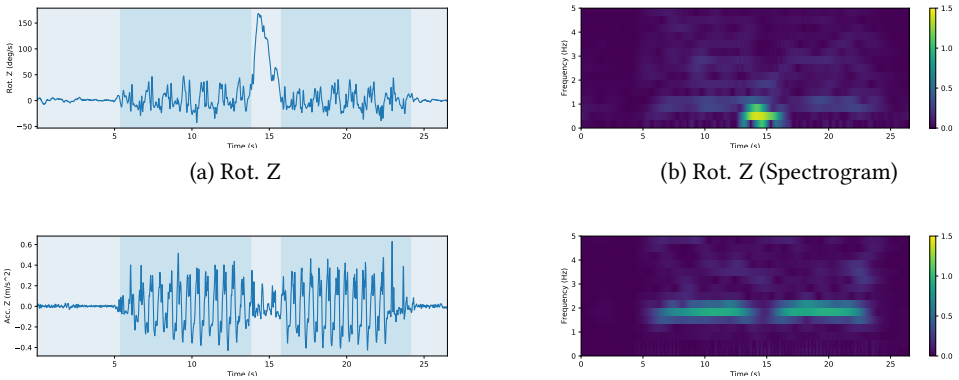


(b) Osteoarthritis patient.

Figure 3.4: Vertical acceleration (m/s^2) of the lower back sensor for two different subjects. Alternating colours mark the consecutive phases: "Stand", "Walk", "Turnaround", "Walk" and "Stop".

Data set description. This data set contains $L = 262$ signals. For this study, we use $d = 2$ dimensions: the angular velocity around the vertical axis ("Rot. Z") and the acceleration in the vertical direction ("Acc. Z"). The segmentations $\mathcal{T}^{(l)}$ are the manual segmentations provided by the medical researchers. Each of them has $K = 4$ change points: "Stand/Walk", "Walk/Turnaround", "Turnaround/Walk" and "Walk/Stop". In the remainder of the manuscript, the time-frequency representation of signals from *Gait* is defined as the absolute value of coefficients of the short-term Fourier transform (STFT), computed with 300 samples per segment and an overlap of 299 samples (to retain the best time resolution possible). Only the $0 - 5$ Hz frequency band, where phenomena of interest are contained, is kept. Change point detection algorithms presented in this work sometimes take as input the stacked amplitudes of the STFTs. In this representation, the signals have $d = 32$ dimensions.

A representative example of a signal and its time-frequency representation is displayed on Figure 3.5. Several comments can be made. The two flat parts at the extremities of "Rot. Z" and "Acc. Z" correspond to the moments when the subject is standing still. The repeated patterns represent the footsteps. The turnaround is especially visible on the angular velocity. In the time-frequency domain, there is a piecewise constant structure. Both "Walk" phases



(a) Rot. Z (b) Rot. Z (Spectrogram)

(c) Acc. Z (d) Acc. Z (Spectrogram)

Figure 3.5: Signal example from *Gait*. The acceleration and rotation on axis (Oz) (time and time-frequency representation) are shown. Alternating colours mark the consecutive phases: “Stand”, “Walk”, “Turnaround”, “Walk” and “Stop”.

are periodic, resulting in energy peaks are visible, around 2 Hz, for the displayed signal. The “Turnaround” regime is a break in the periodicity, as evidenced by the drop in energy at the footstep frequencies, on the time-frequency representation of the acceleration.

2.2.2 Synthetic data sets: *MeanShift* and *FreqShift*

Two synthetic data sets are simulated: *MeanShift* and *FreqShift*. In both situations, simulation parameters are chosen so that the signals approximate the ones from *Gait*. Precisely, *MeanShift*, which contains noisy piecewise constant signals, is an idealized description of the time-frequency representation of *Gait* signals. As for *FreqShift*, which contains noisy piecewise periodic signals, it is an idealized description of *Gait* signals, in both the time domain and the time-frequency domain. Each set is composed of 400 signals, grouped into four subsets of similar characteristics (same length and same noise level).

Simulation strategy. For both data sets, the simulation strategy is the same. Precisely, for a given number K of change points, a number of samples T, a noise level σ, and a signal model (piecewise constant or piecewise periodic), the simulation strategy of a signal is outlined as follows:

1. randomly draw K change point indexes from $\{1, \dots, T\}$ using a Dirichlet distribution (see Remark 3.1),
2. simulate a signal according to the chosen model, using the already drawn change points,
3. randomly draw and add a Gaussian white noise (of standard deviation σ) to each dimension of the signal.

The distribution of the change point locations, displayed on Figure 3.6, is arbitrarily chosen to match the segmentations found in the *Gait* data set.

Remark 3.1 (Dirichlet distribution). *The Dirichlet distribution, denoted* $\mathrm{Dir}(\alpha)$*, is a continuous multivariate probability distribution parametrized by a vector* $\alpha = [\alpha_1, \ldots, \alpha_n]$*. Its support is the set of* $[0,1]^n$*-valued vectors* x *whose coordinates sum to 1, meaning that* $x_1 + \cdots + x_n = 1$*. The probability density function* $f_\alpha(\cdot)$ *is given by*

$$f_\alpha(x_1, \ldots, x_n) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1} \quad (\forall x_i > 0 \text{ s.t. } x_1 + \cdots + x_n = 1) \tag{3.6}$$

where $B(\alpha)$ *is a normalization function. The Dirichlet distribution is used to model the* $K+1$ *regime durations* $t_1/T, (t_2 - t_1)/T, \ldots, (T - t_K)/T$ *associated with a segmentation* $\mathcal{T} = \{t_1, \ldots, t_K\}$ *(durations are normalized by the number of samples* T*). More precisely, to construct* K *random change point indexes, simply draw* $(x_1, \ldots, x_{K+1})$ *from a Dirichlet distribution* $\mathrm{Dir}(\alpha)$*. The change point indexes are given by*

$$t_k = \lfloor T x_1 + \cdots + T x_k \rfloor \tag{3.7}$$

For all synthetic signals, the parameter α *is arbitrarily set to* $(5,5,3,5,1) \times 2000$ *to match the segmentations found in the* Gait *data set.*

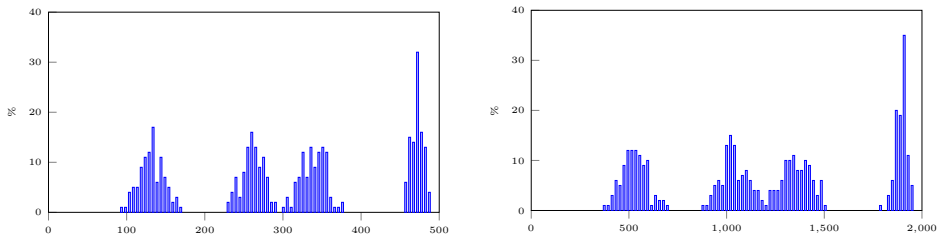
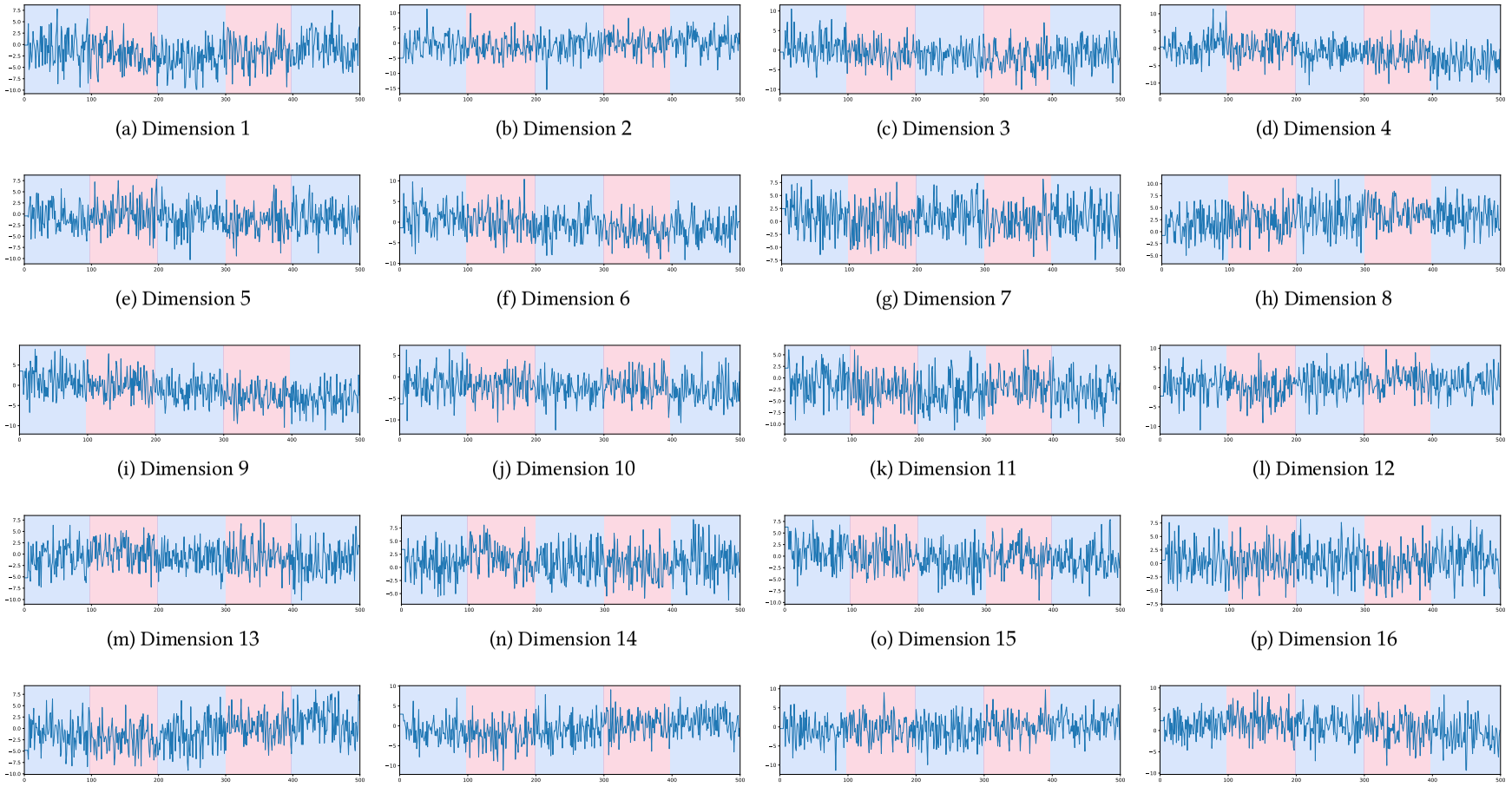


Figure 3.6: Change point repartition for all simulated signals. Left: 500-sample long signals. Right: 2000-sample long signals.

The *MeanShift* data set. The *MeanShift* data set contains $\mathbb{R}^d$-valued ($d = 20$) piecewise constant signals with $K = 4$ change points, $T \in \{500, 2000\}$ samples and a noise level $\sigma \in \{1, 3\}$. We consider four scenarios for different values of (T, σ): Scenario 1, 2, 3 and 4 respectively correspond to (T, σ) equal to $(500, 1)$, $(500, 3)$, $(2000, 1)$ and $(2000, 3)$. For a given scenario, 100 signals are generated according to the following model:

$$y_t = \sum_{k=1}^{K} \delta_k \mathbb{1}(t_k < t) \quad (t = 1, \ldots, T) \tag{3.8}$$

where $\mathcal{T} = \{t_k\}_{k=1}^{K}$ is a random set change indexes, and the $\delta_k \in \mathbb{R}^d$ are such that $\delta_k = [\pm 1, \ldots, \pm 1] \in \mathbb{R}^d$ with random coefficients equal to ± 1. As a result, the complete *MeanShift* data set contains $L = 400$ signals. A signal example is displayed in Figure 3.7.



(a) Dimension 1 (b) Dimension 2 (c) Dimension 3 (d) Dimension 4

(e) Dimension 5 (f) Dimension 6 (g) Dimension 7 (h) Dimension 8

(i) Dimension 9 (j) Dimension 10 (k) Dimension 11 (l) Dimension 12

(m) Dimension 13 (n) Dimension 14 (o) Dimension 15 (p) Dimension 16

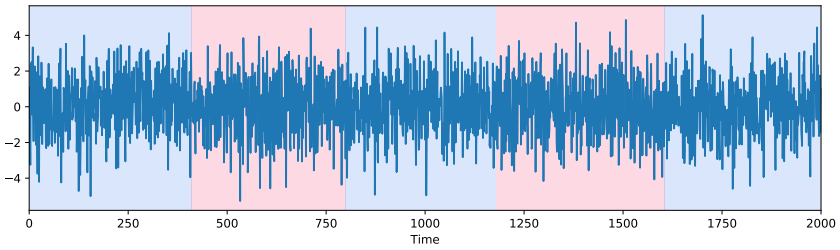
(q) Dimension 17 (r) Dimension 18 (s) Dimension 19 (t) Dimension 20

Figure 3.7: Signal example from *MeanShift* (Scenario 2). The true segmentation is indicated by the alternating colours.

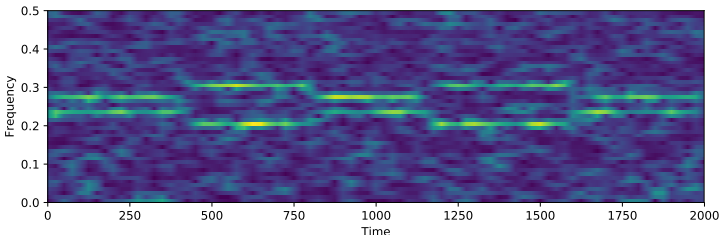
The *FreqShift* data set The *FreqShift* data set contains univariate ($d = 1$) piecewise periodic signals with $K = 4$ change points, $T = 2000$ samples and Signal-to-Noise Ratio SNR $\in \{-5, -1, 0, 2\}$ dB. For each SNR, 100 signals are generated according to the following model:

$$y_t = \sin(2\pi f_1 t) + \sin(2\pi f_2 t) \quad (t = 1, \ldots, T) \tag{3.9}$$

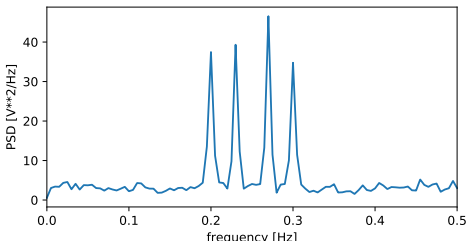
where f_1 and f_2 are frequencies such that vector $[f_1, f_2]$ alternates from $[0.20, 0.30]$ to $[0.23, 0.27]$ at each change point indexes. As a result, the complete *FreqShift* data set contains $L = 400$ signals. In the remainder of the manuscript, the time-frequency representation of signals from *FreqShift* is defined as the absolute value of coefficients of the STFT, computed with 300 samples per segment and 75% overlap. A signal example is displayed on Figure 3.8.



(a) Original signal



(b) Time-frequency representation



(c) Power spectral density

Figure 3.8: Signal example from *FreqShift* (SNR $= -1$ dB). (a): the true segmentation is indicated by the alternating colours.

2.2.3 Comments on the data sets

To evaluate the difficulty of the change point detection task, we can calculate average mean-shift amplitudes across the *MeanShift*, *FreqShift* and *Gait* data sets. For each change

point, the mean-shift amplitude is the difference in mean between the previous regime and the following regime. Intuitively, the larger the amplitude, the easier the task is. Formally, for a change point $t_k^\star$, the mean-shift amplitude Δ and the normalized mean-shift amplitude $\widetilde{\Delta}$ between the sub-signals $z_{\text{left}} := z_{t_{k-1}^\star..t_k^\star}$ and $z_{\text{right}} := z_{t_k^\star..t_{k+1}^\star}$, are respectively given by

$$\Delta(z_{\text{left}}, z_{\text{right}})^2 := \left\|\bar{z}_{\text{left}} - \bar{z}_{\text{right}}\right\|^2 \quad \text{and}$$
$$\widetilde{\Delta}(z_{\text{left}}, z_{\text{right}})^2 := \frac{1}{d}\sum_{i=1}^{d} \frac{(\bar{z}_{\text{left},i} - \bar{z}_{\text{right},i})^2}{\hat{\sigma}^2_{\text{left},i}/T_{\text{left}} + \hat{\sigma}^2_{\text{right},i}/T_{\text{right}}} \tag{3.10}$$

where $\bar{z}_\bullet$, $\bar{z}_{\bullet,i}$, $T_\bullet$ and $\hat{\sigma}^2_{\bullet,i}$ (with $\bullet$ standing for either "left" or "right") respectively are the empirical mean of $z_\bullet$, the empirical mean of $z_{\bullet,i}$ (the i-th dimension of $z_\bullet$), the number of samples of $z_\bullet$ and the (unbiased) empirical standard deviation of $z_{\bullet,i}$. Average mean-shift amplitudes are then computed on the raw signals from *MeanShift* and on the time-frequency representation of the *FreqShift* and *Gait* signals. Several observations can be made from the results provided in Table 3.1.

Scenario	Δ	$\widetilde{\Delta}$
1	4.53	7.24
2	4.82	2.56
3	4.49	14.22
4	4.58	4.83

(a) *MeanShift* data set

SNR	Δ	$\widetilde{\Delta}$
-5 dB	0.51	13.62
-1 dB	0.48	19.36
0 dB	0.48	20.96
2 dB	0.48	24.67

(b) *FreqShift* data set (time-frequency representation)

	Δ	$\widetilde{\Delta}$
Stand/Walk	0.97	50.20
Walk/Turnaround	2.28	31.51
Turnaround/Walk	2.26	30.51
Walk/Stop	0.85	35.21
Average	1.59	36.86

(c) *Gait* data set (time-frequency representation)

Table 3.1: Average mean-shift amplitudes for the *MeanShift*, *FreqShift* and *Gait* data sets.

- **On the *MeanShift* data set**, Scenario 3, Scenario 1, Scenario 4, and Scenario 2 are in ascending order of difficulty, as evidenced by the values of the normalized mean-shift amplitudes. This can be explained by the fact that segmentation is easier with more samples, and less noise. Indeed, the most difficult one, Scenario 2, has the less samples ($T = 500$) and the most noise ($\sigma = 3$).

- **On the *FreqShift* data set**, the SNR and the normalized mean-shift amplitude are positively correlated, meaning that changes are more visible when there are less noise. Comparatively to *MeanShift*, normalized amplitudes are greater. This indicates that change point detection is to be easier on *FreqShift*. Experiments that are presented

later in this manuscript contradict this statement, showing the limits of the mean-shift amplitude as a proxy for segmentation difficulty, when comparing changes of different types.

- **On the *Gait* data set**, the mean-shift amplitudes are different depending on the change point type. For instance, the second and third change points have lower amplitudes, compared to the first and last one. This indicates that "Turnaround" is the most difficult regime to segment. Also, even though the first and last change points separate the same regimes ("Walk" and a rest period), and have comparable mean-shifts, their normalized amplitudes are different. This is due to the fact that the last regime ("Stop") contains a lot less samples than any other regime, on average.

3 Summary tables

The data sets and metrics used in this thesis are summarized in Table 3.2 and Table 3.3.

Data set	T	d	L	noise
MeanShift	500, 2000	20	4×100	$\sigma = 1, 3$
FreqShift	2000	1	4×100	SNR $= -5, -1, 0, 2$ dB
Gait	$1700 - 4000$	2	262	-

Table 3.2: Data set summary table: number of samples T, dimension d, number of signals L, noise level. Note that signals from *FreqShift* and *Gait* also have a time-frequency representation.

Metric	Formula
Hausdorff	$\max\{ \max_{\hat{t}\in\widehat{\mathcal{T}}} \min_{t^\star\in\mathcal{T}^\star} \lvert\hat{t}-t^\star\rvert, \max_{t^\star\in\mathcal{T}^\star} \min_{\hat{t}\in\widehat{\mathcal{T}}} \lvert\hat{t}-t^\star\rvert \}$ (2.1.2)
RandIndex	$\big(\lvert \mathrm{gr}(\widehat{\mathcal{T}}) \cap \mathrm{gr}(\mathcal{T}^\star)\rvert + \lvert \mathrm{ngr}(\widehat{\mathcal{T}}) \cap \mathrm{ngr}(\mathcal{T}^\star)\rvert\big) / T / (T-1)$ (3.2)
Prec	$\lvert \textsc{Tp} \rvert / \widehat{K}$ (3.4)
Rec	$\lvert \textsc{Tp} \rvert / K^\star$ (3.4)
F1 score	$2 \times (\textsc{Prec} \times \textsc{Rec}) / (\textsc{Prec} + \textsc{Rec})$ (3.5)

Table 3.3: Metric summary table: true change point set $\mathcal{T}^\star$, estimated change point set $\widehat{\mathcal{T}}$.

Part II

Greedy change point detection

4

Greedy change point detection

Contents

Abstract

The objective of this chapter is to design a sub-optimal strategy that leads to robust and computationally efficient signal segmentation. To that end, we introduce gCPD (Greedy Change Point Detection), using a sparse regression formulation of the change point detection problem, with an appropriate design matrix. This algorithm greedily approximates the optimal detection solution, using the Orthogonal Matching Pursuit (OMP) strategy. This results in a time complexity of the order of $\mathcal{O}(T)$, where T is the signal length. In addition, asymptotic consistency results are derived.

1 Statistical model for change point detection

1.1 Problem formulation

We consider the segmentation of an $\mathbb{R}^d$-valued noisy signal with $K^\star$ change points:

$$y_t = u_t + \varepsilon_t \quad (t = 1, \dots, T) \tag{M4}$$

where u_t is a $\mathbb{R}^d$-valued deterministic piecewise constant signal and $(\varepsilon_t)_{0<t\leq T}$ are zero-mean i.i.d. random variables. Without loss of generality, each dimension of u is assumed to have zero mean. Let $\mathcal{T}^\star := \{t_k^\star \mid k = 1, \dots, K^\star\}$ denote the set of change point indexes $(t_1^\star < t_2^\star < \cdots < t_{K^\star}^\star)$; define in addition the dummy indexes $t_0^\star := 0$ and $t_{K^\star+1}^\star := T$. Additional technical assumptions on u_t and ε_t are described later. In matrix form, the noisy piecewise constant model (M4) of the observed signal y is rewritten as follows:

$$Y = U + E \tag{4.1}$$

where Y is the $T \times d$ matrix containing the observed signal $\{y_t\}_t$, U is the $T \times d$ matrix containing the underlying piecewise constant signal $\{u_t\}_t$, E is $T \times d$ matrix containing the noise realization $\{\varepsilon_t\}_t$. Our objective is to design a detection algorithm that locates change points with precision, while being computationally fast. We compare our approach to several methods described in Chapter 2, among which `Opt` is the most precise one and `Win` is the fastest. The algorithm is expected to address both known and unknown $K^\star$ and to be asymptotically consistent, in the sense of Definition 2.1 on page 52. This asymptotic setting is formalized in Chapter 2, in particular Section 1.3 on page 52. The definition of asymptotic consistency is simply recalled below.

Definition 4.1 (Asymptotic consistency)**.** *A change point detection algorithm is said to be asymptotically consistent if the estimated segmentation $\widehat{\mathcal{T}} = \{\hat{t}_1, \hat{t}_2, \dots\}$ satisfies the following conditions, when $T \longrightarrow +\infty$:*

(i) $P(|\widehat{\mathcal{T}}| = K^\star) \longrightarrow 1$,

(ii) $\frac{1}{T}\left\|\widehat{\mathcal{T}} - \mathcal{T}^\star\right\|_\infty \xrightarrow{p} 0$,

where the distance between two change point sets is defined by

$$\left\|\widehat{\mathcal{T}} - \mathcal{T}^\star\right\|_\infty := \max\left\{ \max_{\hat{t}\in\widehat{\mathcal{T}}} \min_{t^\star\in\mathcal{T}^\star} |\hat{t} - t^\star|,\ \max_{t^\star\in\mathcal{T}^\star} \min_{\hat{t}\in\widehat{\mathcal{T}}} |\hat{t} - t^\star| \right\}. \tag{4.2}$$

1.2 Related work

As described in detail in Section 3 on page 63, several methods already exist to tackle this change point detection problem, namely `Opt`, `BinSeg`, `BotUp` and `Win`, each combined with the c_{L_2} cost function.

- The most accurate but also computationally intensive detection algorithm is `Opt`. It can only be applied if the number of changes is known beforehand. Its complexity is of the order of $\mathcal{O}(KT^2)$ (where T is the number of samples and $K^\star$ the number of changes).

- `Win` is a fast approximate method that searches change points locally and can accommodate known and unknown number of change points.

- Binary segmentation `BinSeg` and bottom-up segmentation `BotUp` are sequential tree-based methods that address several of the drawbacks of `Opt` and `Win`. They are able to address both known and unknown $K^\star$ and are faster than `Opt`. However they remain local methods, and their estimation is not as optimal as global methods [111].

In order to create efficient procedures that use the whole signal to detect change points, relations between the change point detection problem and sparse regression, with an appropriate design matrix, have been investigated in the literature. Generally, methods to approximate the solution of sparse regression fall into two categories: basis pursuit and matching pursuit. Methods based on basis pursuit have been applied to detect an *unknown* number of change points, for instance regressions with a total variation penalty (or a fused lasso penalty) [73, 160]. Their implementations are efficient, with complexity of the order of $\mathcal{O}(KT)$ or $\mathcal{O}(T \log T)$ [160] and there are theoretical guarantees of detecting correct changes. Nevertheless, the number of change points is treated as unknown and cannot be set explicitly. Conversely, we show in this chapter that methods based on matching pursuit can accommodate either $K^\star$ *known or* $K^\star$ *unknown*. In a nutshell, they are greedy approaches that produce a sequence of incremental approximations of the original signal, using a set of elementary signals, called atoms. The most well-known examples of such algorithms are Matching Pursuit (MP) [54] and its extension, Orthogonal Matching Pursuit (OMP) [152]. Roughly, MP constructs the approximation by sequentially adding the projection on a new atom. OMP follows a slightly different strategy: the approximation is the orthogonal projection of the original signal onto a linear subspace spanned by an increasing number of atoms. While MP is faster than OMP from a computational standpoint, its convergence rate can be slow, because the same atom can be selected several times [54, 152]. Those strategies are often used in contexts like compressed sensing and sparse approximation [38, 53, 54, 152], but not on change point detection problems.

1.3 Contributions of the chapter

We propose a sequential approach, called gCPD for "greedy change point detection", that sequentially generates change point estimates $\hat{t}^{(k)}$ (at the k-th iteration) and removes the associated mean-shifts from the initial signal, until a stopping criterion is met. (In the following, the subscript $\cdot^{(k)}$ refers to the k-th iteration of a sequential algorithm.) Our algorithm is easily implemented, and, as we demonstrate in this work, is asymptotically consistent. We further show that, in practice, this method can accommodate a known or unknown $K^\star$, and has linear complexity.

2 Change point detection as a sparse regression task

This section presents the equivalence between change point detection and sparse regression, then describes the implementation of our algorithm. To that end, we start off by introducing the atoms and dictionary.

2.1 The Heaviside decomposition

Using the sparse approximation terminology, we introduce a matrix $S \in \mathbb{R}^{T \times T-1}$, called "dictionary", whose columns are well-suited elementary signals $S^\alpha \in \mathbb{R}^T$ (in matrix form), called "atoms". The continuous function from which all atoms are sampled is given by

$$\forall \tau \in [0,1] \quad h^\alpha(\tau) := -\sqrt{\frac{1-\alpha}{\alpha}} \mathbb{1}\{\tau \leq \alpha\} + \sqrt{\frac{\alpha}{1-\alpha}} \mathbb{1}\{\alpha < \tau\}. \tag{4.3}$$

It is a centered and scaled step function with a single change point located at α.

Definition 4.2 (Atoms and dictionary). *The atoms $S^\alpha \in \mathbb{R}^T$ are univariate signals (in matrix form) and defined as follows:*

$$S^\alpha := \frac{1}{\sqrt{T}} \left[h^\alpha\left(t/T\right) \right]_{1 \leq t \leq T}. \tag{4.4}$$

The dictionary matrix $S \in \mathbb{R}^{T \times T-1}$ is defined as the concatenation of $T-1$ atoms:

$$S := [S^{1/T}, S^{2/T}, \ldots, S^{1-1/T}]. \tag{4.5}$$

The elementary signal S^α has zero mean and is scaled to unit Euclidean norm. It is easy to see that the dictionary matrix S has full rank: its columns form a basis of the subspace spanned by signals with zero mean. Figure 4.1 displays an atom example. Those atoms are

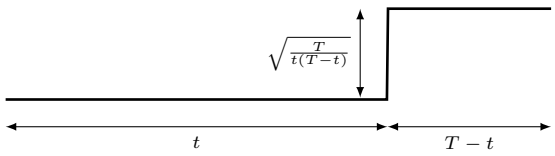


Figure 4.1: Atom example $S^{t/T}$ with $t \in \{1, \ldots, T\}$.

useful because any signal can be expressed as a linear combination of those elementary signals. More importantly, any piecewise constant signal is a linear combination of only a few of those atoms. This decomposition is called here the Heaviside decomposition, after the fact that each elementary signal S^α is a translated and scaled version of the Heaviside step function. Similar decompositions have been used in [65, 73, 160]. In detail, let $x = \{x_t\}_t$ denote an $\mathbb{R}^d$-valued signal with T samples and zero mean. Let $X \in \mathbb{R}^{T \times d}$ be the matrix representation of x, i.e. $X_{t,\bullet} := x_t$. Then there exists a unique $\Delta \in \mathbb{R}^{T-1 \times d}$ such that

$$X = S\Delta \tag{4.6}$$

where S is defined in (4.5). Furthermore, some algebraic manipulations give

$$\Delta_{t,\bullet} = \sqrt{\frac{t(T-t)}{T}} \left(X_{t+1,\bullet} - X_{t,\bullet}\right). \tag{4.7}$$

The support of Δ (the set of non-zero coefficients) is exactly the set of change points of the signal X, meaning that

$$t \text{ is a change point} \iff x_t \neq x_{t+1} \iff \Delta_{t,\bullet} \neq 0. \tag{4.8}$$

The decomposition (4.6) is called the Heaviside decomposition.

2.2 Equivalence to a sparse regression task

In matrix form, the piecewise constant model (M4) of the observed signal y is rewritten using the Heaviside decomposition:

$$Y = S\Delta^\star + E \tag{4.9}$$

where $\Delta^\star \in \mathbb{R}^{T-1\times d}$ is the Heaviside representation of the unobserved piecewise constant signal U. Estimating the set of indexes $\mathcal{T}$ from the observed signal y and solving the sparse regression problem (4.9) are two equivalent tasks. In addition, $\Delta^\star$ is the (sparse) matrix such that

$$\forall t^\star = t_1^\star, \ldots, t_{K^\star}^\star, \quad \Delta_{t^\star,\bullet}^\star = \frac{1}{\sqrt{T}}\sqrt{t^\star(T-t^\star)}\,(U_{t^\star+1,\bullet} - U_{t^\star,\bullet}) \quad \text{and 0 elsewhere.} \tag{4.10}$$

Interestingly, both the amplitude and location of a mean-shift influence the corresponding coefficient in the Heaviside representation. Figure 4.2 displays a signal example and its representation. The set of breakpoints $\mathcal{T}$ is directly related the support of $\Delta^\star$:

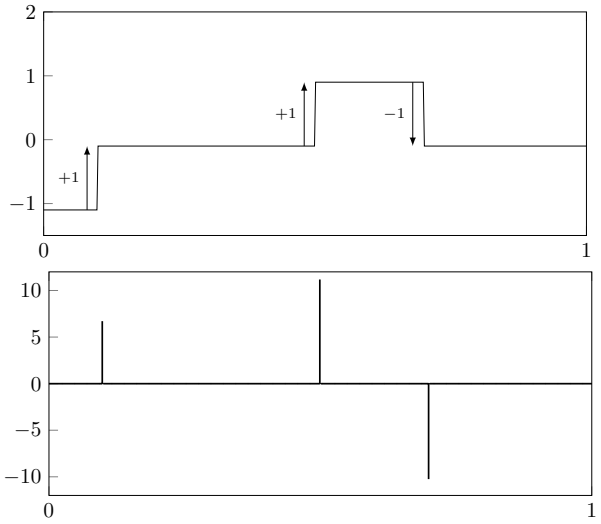


Figure 4.2: Top: piecewise constant signal example ($T = 500$). Bottom: corresponding Heaviside representation. The x-axis is t/T.

$$\mathcal{T} = \{t \text{ s.t. } \Delta_{t,\bullet}^\star \neq 0\} \tag{4.11}$$

In particular, the number of breakpoints is easily recovered from the matrix $\Delta^\star$:

$$\|\Delta^\star\|_{0,1} = \sum_{t=1}^{T-1} \mathbb{1}(\Delta_{t,\bullet}^\star \neq 0) = |\mathcal{T}^\star| = K^\star. \tag{4.12}$$

2.3 Greedy change point detection: the gCPD algorithm

Since the change point detection problem can be seen as a sparse regression problem (4.9), we propose to solve it using the OMP principle [54, 152] with the Heaviside dictionary. In

the following, our algorithm is referred to as gCPD for "greedy change point detection" and outlined in Algorithm 4.1. The algorithm is an iterative procedure searching for a solution of (4.9) by sequentially generating change point estimates, and removing the associated mean-shifts from the initial signal, until a stopping criterion is met.

Initialization. At iteration $k = 1$, gCPD starts by searching the atom $S^{t/T}$ ($t = 1, \dots, T-1$) that is most correlated with the signal Y. The first change estimate $\hat{t}^{(1)}$ of gCPD corresponds to the atom $S^{t/T}$ such that $\left\|Y' S^{t/T}\right\|^2$ is maximal:

$$\hat{t}^{(1)} := \underset{t<T}{\arg\max} \left\|Y' S^{t/T}\right\|^2. \tag{4.13}$$

Then Y is projected on the subspace orthogonal to the selected atom $S^{\hat{t}^{(1)}/T}$; the resulting residual is denoted $\hat{R}^{(1)}$:

$$\widehat{R}^{(1)} := Y - \widehat{P}^{(1)} Y. \tag{4.14}$$

where the orthogonal projection $\widehat{P}^{(1)}$ on the selected atom is defined by

$$\widehat{P}^{(1)} := S^{\hat{t}^{(1)}/T} (S^{\hat{t}^{(1)}/T})'. \tag{4.15}$$

Iteration. After k iterations ($k \geq 1$), the set of already estimated indexes is denoted $\widehat{\mathcal{T}}^{(k-1)} := \{\hat{t}^{(1)}, \dots, \hat{t}^{(k-1)}\}$. The k-th change point estimate $\hat{t}^{(k)}$ is given by

$$\hat{t}^{(k)} := \underset{t<T}{\arg\max} \left\|(\hat{R}^{(k-1)})' S^{t/T}\right\|^2. \tag{4.16}$$

The orthogonal projection $\widehat{P}^{(k)}$ on the selected atoms at iteration k is defined by

$$\widehat{P}^{(k)} := S(\widehat{\mathcal{T}}^{(k)}) S(\widehat{\mathcal{T}}^{(k)})^{\dagger} \tag{4.17}$$

where $S(\widehat{\mathcal{T}}^{(k)})$ denotes the sub-matrix of S containing the columns of the already chosen atoms $[S^{\hat{t}^{(1)}/T}, \dots, S^{\hat{t}^{(k)}/T}]$. The residual signal $\hat{R}^{(k)}$ is

$$\widehat{R}^{(k)} := Y - \widehat{P}^{(k)} Y. \tag{4.18}$$

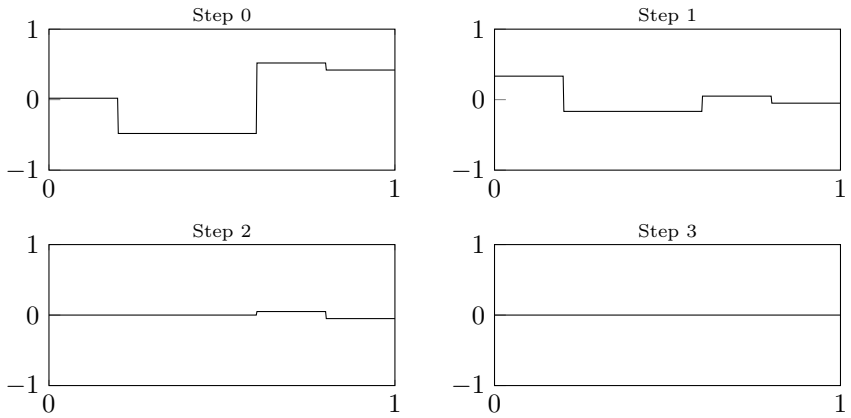


Figure 4.3: Example of a noiseless piecewise constant signal ($T = 500$) and the successive residual signals when gCPD is applied. At step 0, the initial signal is displayed.

The algorithm gCPD is illustrated on Figure 4.3. At Step 0, the original signal is displayed. At first, the second change point is selected because it is the most obvious one, that is to say that the correlation (4.19) with the associated Heaviside atom is the greatest. Then the projection "deletes" this breakpoint. Note that the mean-shift has not entirely disappeared but was greatly reduced. This is because the dictionary is not orthogonal. Atoms with breakpoints close to each other are heavily correlated. In the sparse approximation literature, such a dictionary is called "coherent" [38]. As a result, a local shift remains. However the mean value left of the selected change point is exactly equal to the right mean value (4.19). Therefore the global contribution of the chosen atom to the residual is zero. Interestingly, the other change points are left untouched by the projection, both in location and amplitude. After this step, another breakpoint is selected and its contribution removed. After three steps, there is no signal left, meaning that the change point locations and amplitudes have been retrieved.

2.4 Heuristics for gCPD

Some comments can be made on both the greedy selection and the projection to provide some intuition about how gCPD operates. The greedy selection (4.16) selects the most pronounced mean-shift present in the signal. To illustrate, let $X \in \mathbb{R}^{T\times d}$ denote a signal (in matrix form) and $\tau = t/T$ for a certain index t between 1 and $T-1$. The correlation between the atom S^τ and X is

$$\begin{aligned} \|X'S^\tau\|^2 &= \sum_{j=1}^{d} \left\langle S^\tau \middle| X_{\bullet,j} \right\rangle^2 \\ &= \sum_{j=1}^{d} \frac{1}{T}\Big(-\sqrt{\frac{1-\tau}{\tau}} \sum_{s=1}^{t} X_{s,j} + \sqrt{\frac{\tau}{1-\tau}} \sum_{s=t+1}^{T} X_{s,j} \Big)^2 \\ &= T\tau(1-\tau) \left\| \overline{X}_{0..t} - \overline{X}_{t..T} \right\|^2 \end{aligned} \tag{4.19}$$

where $\overline{X}_{0..t}$ and $\overline{X}_{t..T}$ are respectively the empirical means of $\{X_{s,\bullet}\}_{s\leq t}$ and $\{X_{s,\bullet}\}_{s>t}$. The correlation (4.19) between a signal X and an elementary signal S^τ is proportional to the squared standard t-test for the comparison of two means, which is common expression in the context of single change point detection [29, 31]. In other words, the selection step selects the most pronounced change point. As shown in (4.19), a break is more likely to be chosen if the associated mean-shift amplitude is large and if its location is away from the edges of the signal. The quantity $\tau(1-\tau)$ is maximal in the middle of the signal, meaning that breaks at this location are more visible. The top-down algorithm `BinSeg` also uses this greedy selection. Therefore, the first detected change point is the same for both `BinSeg` and gCPD. However, `BinSeg` recursively splits the signal after, while gCPD performs an orthogonal projection (4.17). By applying $\widehat{P}^{(k)}$ to Y, the contribution of the already selected change points $\widehat{\mathcal{T}}^{(k)}$ is removed from the signal. Indeed, the projected signal $\widehat{P}^{(k)}Y$ is equal to the best approximation by a piecewise constant signal formed with the selected atoms and is subtracted from the initial data, yielding the residual (4.18). Interestingly, a selected atom cannot be selected again in a subsequent iteration because the residual is orthogonal to all of the previously selected atoms, thus setting the correlation to 0. In other words, thanks to the OMP principle, we cannot detect the same change point twice.

Algorithm 4.1 gCPD

Input: centered data Y, stopping criterion.
Initialize $\widehat{R} \leftarrow Y$, $\widehat{\mathcal{T}} \leftarrow \{\}$ ▷ Residual and set of breakpoints
while stopping criterion is not met **do**
  Set $\hat{t} \leftarrow 1$, $r \leftarrow \widehat{R}_{1,\bullet}$, $m^2 \leftarrow \frac{T}{T-1}\|r\|^2$. ▷ Variable $\hat{t}$ holds the change point estimate.
  for $t = 2, \dots, T-1$ **do**
    if $\frac{T}{t(T-t)}\left\|r + \widehat{R}_{t,\bullet}\right\|^2 > m^2$ **then** ▷ Variable selection.
      $\hat{t} \leftarrow t$
      $m^2 \leftarrow \frac{T}{t(T-t)}\left\|r + \widehat{R}_{t,\bullet}\right\|^2$
    end if
    $r \leftarrow r + \widehat{R}_{t,\bullet}$
  end for
  Add the change point estimate $\hat{t}$ to the set of selected breakpoints:

$$\widehat{\mathcal{T}} \leftarrow \widehat{\mathcal{T}} \cup \{\hat{t}\}. \tag{4.20}$$

  Let $\widehat{P}$ denote the orthogonal projection onto the subspace spanned by the selected columns of S:

$$\widehat{P} \leftarrow S(\widehat{\mathcal{T}})S(\widehat{\mathcal{T}})^{\dagger}. \tag{4.21}$$

  Update the residual

$$\widehat{R} \leftarrow Y - \widehat{P}Y. \tag{4.22}$$

end while
Output: set $\widehat{\mathcal{T}}$ of change point estimates.

2.5 Complexity analysis

Generally, matching pursuit algorithms have $\mathcal{O}(dT^2)$ complexity at each step [54]. The cost is mainly driven by the variable selection step, for which $T-1$ correlations between a signal of size $T \times d$ and an atom of size T are needed. In the context of change point detection, we describe a significant speed-up to compute the correlations which yields a $\mathcal{O}(KdT)$ complexity (if K iterations are performed). Indeed, since every dimension of the residuals has zero mean, the following equality holds:

$$\begin{aligned}
\overline{X}_{t..T} - \overline{X}_{0..t} &= \frac{1}{T-t}\sum_{s=t+1}^{T} X_{s,\bullet} - \frac{1}{t}\sum_{s=1}^{t} X_{s,\bullet} \\
&= \frac{1}{T-t}\sum_{s=1}^{t} -X_{s,\bullet} - \frac{1}{t}\sum_{s=1}^{t} X_{s,\bullet} \\
&= -\frac{T}{t(T-t)}\sum_{s=1}^{t} X_{s,\bullet}
\end{aligned} \tag{4.23}$$

where X can be replaced by any residual signal $\widehat{R}^{(k)}$, t is any index between 1 and $T-1$, $\overline{X}_{0..t}$ and $\overline{X}_{t..T}$ are defined as in (4.19). Hence, the largest correlation $\|X'S^{\tau}\|^2$ can be found incrementally using a cumulative sum and keeping track of the current maximum. The variable selection is now linear in the number of samples and the dimension. Applying

the orthogonal projection is equivalent to inverting a matrix of size k (at step k) and then a matrix multiplication. The resulting computational is also linear. The same goes for the residual update. Therefore the complexity of one iteration of the algorithm is $\mathcal{O}(dT)$. Overall, the complexity of gCPD is $\mathcal{O}(KdT)$.

2.6 Stopping criterion

A crucial element of the gCPD algorithm is the stopping criterion. The choice and calibration of a stopping rule is closely related to the issue of finding the number of change points in a signal. If the number of change points $K^\star$ is known, one simply stops the algorithm after $K^\star$ iterations. If it is unknown, a linear penalty [82, 112] can be added to yield a the following optimization problem:

$$\min_{\Delta \in \mathbb{R}^{(T-1)\times d}} \|Y - S\Delta\|^2 + \beta \|\Delta\|_{0,1} \quad (\beta > 0) \tag{4.24}$$

where $\beta > 0$ is the smoothing parameter and

$$\|\Delta\|_{0,1} = \sum_{t=1}^{T-1} \mathbb{1}(\Delta_{t,\bullet} \neq 0) \tag{4.25}$$

is the number of non-zero rows of Δ. An adapted stopping rule to approximate the linearly penalized change point detection is to stop at the k-th iteration if

$$\left\|\widehat{R}^{(k-1)}\right\|^2 - \left\|\widehat{R}^{(k)}\right\|^2 < \beta. \tag{4.26}$$

3 Conclusion

In this chapter, a sequential approach, called gCPD, for change point detection is described that takes advantage of this formulation. In practice, gCPD can accommodate a known or unknown K, and has linear complexity. In Appendix, this algorithm is shown to produce asymptotically consistent estimates. Numerical comparison of gCPD to standard approximate and optimal procedures is carried out in Chapter 6. An interesting issue for future research is to derive faster convergence rates for the change point estimates. As a comparison, optimal procedures from the literature achieve a convergence speed of the order of $1/T$ [110].

Appendices

4.A Theoretical Analysis

This section presents a theoretical study of gCPD. In particular, a result of asymptotic consistency is obtained.

4.A.1 Model and technical assumptions

In order to establish our main result, several assumptions on the signal U and the noise E are made.

Assumption 4.1. *A positive constant M exists such that*

$$\max_{1\leq t\leq T} \|U_t\| \leq M < +\infty. \tag{4.27}$$

Assumption 4.2. *The change points $t_1^\star, \ldots, t_{K^\star}^\star$ satisfy*

$$\forall 0 \leq k \leq K^\star, \quad |t_{k+1}^\star - t_k^\star|/T \geq \underline{\Delta}, \tag{4.28}$$

where $\underline{\Delta}$ is a positive constant such that $0 < \underline{\Delta} < 1$.

In Assumption 4.2, the spacing between change point *fractions* is bounded away from zero. This prevents breakpoints to be too close to each other and become indistinguishable. In the literature, change point fractions are sometimes allowed to slowly tend to zero, as T grows to infinity (for instance at the rate of $\ln(T)^2/T$ in [73]). As an example, both Assumption 4.1 and Assumption 4.2 are satisfied when the signal U is sampled from a fixed piecewise constant function $f : [0,1] \longrightarrow \mathbb{R}^d$, with sampling frequency $1/T$.

Assumption 4.3. *The $\mathbb{R}^d$-valued random sequence $(\varepsilon_t)_t$ is i.i.d. isotropic Gaussian with mean zero and variance σ^2.*

Our theoretical analysis relies heavily on Assumption 4.3. In a nutshell, since Gaussian noise concentrates around its mean, therefore, with high probability, segmenting a noisy piecewise constant signal is not harder than segmenting a noiseless piecewise constant signal. Here, the Gaussian distribution is assumed, for technical convenience. It is reasonable to conjecture that our main result remains valid under a weakened assumption, in particular with sub-Gaussian distributions, as in [35, 73].
In the following, it is important to measure how close the elements of $\widehat{T}$ and $T^\star$ are, even though the estimated number of change points is different from the true number. To that end, define the quantity $d(A|B)$ between two sets A and B by

$$d(A|B) := \sup_{b\in B} \inf_{a\in A} |a-b|. \tag{4.29}$$

This "distance" measures how close to the elements of A all the elements of B are. The Hausdorff distance can easily be rewritten using this $d(\cdot|\cdot)$:

$$\textsc{Hausdorff}(\mathcal{T}^\star, \widehat{\mathcal{T}}) = \max[d(\mathcal{T}^\star|\widehat{\mathcal{T}}), d(\widehat{\mathcal{T}}|\mathcal{T}^\star)]. \tag{4.30}$$

4.A.2 Asymptotic consistency

Intuitively, the true change point fractions $t_k^\star/T$ are estimated more and more precisely as the number of samples increases. This result is formally stated by Theorem 4.1.

Theorem 4.1. *Let Y follow Model M4, and suppose that Assumption 4.1, Assumption 4.2 and Assumption 4.3 hold. Let $\widehat{\mathcal{T}}$ denote the set of estimated change points after $k \leq K^\star$ steps of* gCPD. *Then there exist positive constants C_1, C_2 such that $P(\mathcal{A}_T) \geq 1 - C_1/T$ where*

$$\mathcal{A}_T := \{\frac{1}{T} d(\mathcal{T}^\star | \widehat{\mathcal{T}}) \leq C_2 \gamma_T\} \tag{4.31}$$

with $(\gamma_T)_T$ a non-increasing and positive sequence tending to zero as T tends to infinity and satisfying $\gamma_T \sqrt{T/\ln(T)} \longrightarrow +\infty$.

Corollary 4.1 (Consistency of gCPD)**.** *Algorithm* gCPD *(Algorithm 4.1), stopped after $K^\star$ iterations, is asymptotically consistent (in the sense of Definition 2.1).*

Corollary 4.1 means that gCPD produces consistent estimates of the change point fractions of the signal U. This is true as long as the iterations stop before all true breakpoints are detected. In other words, only the first $K^\star$ rounds of Algorithm 4.1 are meaningful. In the situation where $k > K^\star$, meaning that all change points have been selected, the orthogonal projection "deletes" all breaks and the correlation between the residual and the atoms is only driven by noise.
In the sparse approximation literature, guarantees of perfect reconstruction of the sparse solution (which in our context is equivalent to detecting the correct change point indexes) have been obtained under different conditions. They most notably include the Mutual Incoherence Property (MIP) [38], the Restricted Isometry Property (RIP) [53] and the Exact Recovery Condition (ERC) [152]. However, the atoms of our dictionary are too heavily correlated with each other to satisfy any of those conditions. For instance, consider the MIP condition, which can be formulated as follows in our context:

$$\text{The MIP condition is verified} \quad \Leftrightarrow \quad \max_{s,t} \left| \left\langle S^{s/t} \middle| S^{t/T} \right\rangle \right| < 1/(2K^\star - 1) \tag{4.32}$$

Under this condition, it has been shown that the estimated change points are equal to the true change points, with high probability [38, Theorem 7]. Nevertheless, after simple algebraic manipulations, it is easy to show that $\max_{s,t} \left| \left\langle S^{s/t} \middle| S^{t/T} \right\rangle \right|$ tends to one as T grows to infinity, and therefore, is larger than $1/2$ for T large enough. As a result, when $K^\star > 1$, meaning that there are more than one change, the MIP is not satisfied. To prove the asymptotic consistency of the proposed algorithm, the usual theoretical analysis of the OMP literature cannot be applied.

4.A.3 Sketch of proof of Theorem 4.1

We give here the outline of the proof of Theorem 4.1. In this section, the following notations are used. Let $\widehat{\mathcal{T}} = \{\hat{t}_k | k = 1, \ldots, \widehat{K}\}$ be the set of $\widehat{K}$ estimated by gCPD after $\widehat{K} < K^\star$ iterations ($\hat{t}_1 < \hat{t}_2 < \cdots < \hat{t}_{\widehat{K}}$); define in addition the dummy indexes $\hat{t}_0 := 0$ and $\hat{t}_{\widehat{K}+1} := T$. Denote by $\widetilde{\mathcal{T}}$ the set of true change points detected by $\widehat{\mathcal{T}}$, up to a distance $TC_2\gamma_T$, where C_2 and γ_T are introduced in Theorem 4.1:

$$\widetilde{\mathcal{T}} := \{t^\star \in \mathcal{T}^\star \text{ s.t. } \exists \hat{t} \in \widehat{\mathcal{T}}, |t^\star - \hat{t}| \leq TC_2\gamma_T\}. \tag{4.33}$$

Let $\widehat{P}$ and $\widetilde{P}$ denote the orthogonal projections on the estimated change points and on the detection change points respectively:

$$\widehat{P} := S(\widehat{\mathcal{T}}) S(\widehat{\mathcal{T}})^\dagger \quad \text{and} \quad \widetilde{P} := S(\widetilde{\mathcal{T}}) S(\widetilde{\mathcal{T}})^\dagger. \tag{4.34}$$

In addition, we define: $\forall \tau \in (0,1)$,

$$\begin{aligned}
\hat{\phi}_Y(\tau) &:= \frac{1}{\sqrt{T}} \left\| (Y - \widehat{P}Y)' S^\tau \right\| \quad \text{and} \quad \hat{\phi}_Y(0) = \hat{\phi}_Y(1) = 0, \\
\hat{\phi}_U(\tau) &:= \frac{1}{\sqrt{T}} \left\| (U - \widehat{P}U)' S^\tau \right\| \quad \text{and} \quad \hat{\phi}_U(0) = \hat{\phi}_U(1) = 0, \\
\hat{\phi}_E(\tau) &:= \frac{1}{\sqrt{T}} \left\| (E - \widehat{P}E)' S^\tau \right\| \quad \text{and} \quad \hat{\phi}_E(0) = \hat{\phi}_E(1) = 0, \\
\tilde{\phi}_U(\tau) &:= \frac{1}{\sqrt{T}} \left\| (U - \widetilde{P}U)' S^\tau \right\| \quad \text{and} \quad \tilde{\phi}_U(0) = \tilde{\phi}_U(1) = 0.
\end{aligned} \tag{4.35}$$

Note that the change point gCPD selects at the next iteration is the index $\hat{t}$ given by

$$\hat{t} = \underset{t=1,\dots,T}{\arg\max}\ \hat{\phi}_Y(t/T). \tag{4.36}$$

In addition, introduce the event $\mathcal{B}_T$:

$$\mathcal{B}_T := \{ \sup_{1 \le t_1 < t_2 \le T} \frac{1}{\sqrt{t_2 - t_1}} \left\| \sum_{t=t_1+1}^{t_2} E_{t,\bullet} \right\| \le \ln(T) \} \tag{4.37}$$

Remark 4.1 (Informal sketch of proof)**.** *Before stating the successive lemmas that form the proof, its main arguments are informally presented. The objective is to prove that if* gCPD *has correctly detected true change points (meaning that the estimates are no further than $TC\gamma_T$ from a true change index), then the next one will also be close to a true change, provided there are still changes to detect. It is first shown that the noise is well-behaved: on the event $\mathcal{B}_T$, whose probability tends to one, the noise can be bounded. On this event $\mathcal{B}_T$,* gCPD *is then treated deterministically in the proof and the three following arguments are made:*

- *The correlation to maximize, $\hat{\phi}_Y$, is uniformly close to the one without noise, $\hat{\phi}_U$. (Alternatively, $\hat{\phi}_E$ is uniformly close to zero.)*
- *The noiseless correlation $\hat{\phi}_U$ is uniformly close to $\tilde{\phi}_U$, because estimated change points are close to true ones.*
- *The maximum of $\tilde{\phi}_U$ is reached on a true undetected change point $t^\star \in \mathcal{T}^\star \backslash \widetilde{\mathcal{T}}$.*

By combining those three arguments for each iteration of gCPD*, until $\widehat{K} = K^\star$, Theorem 4.1 is proven. The consistency of the algorithm is a straightforward application of Theorem 4.1.*

In the following lemma, the event $\mathcal{B}_T$ is shown to have high probability.

Lemma 4.1. *Let Assumption 4.3 hold. Then the event $\mathcal{B}_T$ satisfies $P(\mathcal{B}_T) \ge 1 - C_3/T$, where C_3 is a positive constant.*

Thanks to Lemma 4.1, the correlation $\hat{\phi}_E$ is shown to uniformly close to zero.

Lemma 4.2. *Let Assumption 4.3 hold. Then on the event $\mathcal{B}_T$, which has probability larger than $1 - C_3/T$, the following occurs:*

$$\sup_\tau\ \hat{\phi}_E(\tau) \le 2\sqrt{\frac{\ln(T)}{T}}. \tag{4.38}$$

The following lemma considers the correlation $\tilde{\phi}_U$, or in other words, the situation without noise where all estimated change points are *equal* to true changes.

Lemma 4.3. *Let U follow Model M4 and suppose that Assumption 4.2 holds. Then there exists $t^\star \in \mathcal{T}^\star \backslash \widetilde{\mathcal{T}}$ such that*

$$t^\star = \arg\max_t \widetilde{\phi}_U(t/T). \tag{4.39}$$

In addition, the following holds:

$$\forall t = 1, \dots, T, \quad \widetilde{\phi}_U(t^\star/T) - \widetilde{\phi}_U(t/T) \geq C_5 |t - t^\star|/T \tag{4.40}$$

where C_5 is a positive constant.

If the estimated change points are assumed to be close to the true ones, the projections of U on $\widehat{\mathcal{T}}$ and $\widetilde{\mathcal{T}}$ must be close, leading to Lemma 4.4.

Lemma 4.4. *Let U follow Model M4, and suppose that Assumption 4.1 and Assumption 4.2 hold. Further assume that*

$$\frac{1}{T} d(\mathcal{T}^\star | \widehat{\mathcal{T}}) \leq C_2 \gamma_T \tag{4.41}$$

where γ_T and C_2 are introduced in Theorem 4.1. Then the following holds:

$$\sup_{\tau \in [0,1]]} |\hat{\phi}_U(\tau) - \widetilde{\phi}_U(\tau)| \leq C_6 \gamma_T / \sqrt{T} \tag{4.42}$$

where C_6 is a positive constant.

By combining Lemmas 4.2 to 4.4, the next estimate $\hat{t}$ of gCPD is shown to be at a distance no larger than $TC_2\gamma_T$ from a true change point, assuming that all previous estimates are also that close, and there still changes to detect. This is formally stated in Lemma 4.5.

Lemma 4.5. *Let Y follow Model M4, and suppose that Assumption 4.1, Assumption 4.2 and Assumption 4.3 hold. Further assume that*

$$\frac{1}{T} d(\mathcal{T}^\star | \widehat{\mathcal{T}}) \leq C_2 \gamma_T \tag{4.43}$$

where γ_T and C_2 are introduced in Theorem 4.1. Then on the event $\mathcal{B}_T$, which has probability larger than $1 - C_3/T$, the following occurs:

$$\frac{1}{T} d(\mathcal{T}^\star | \{\hat{t}\} \cup \widehat{\mathcal{T}}) \leq C_2 \gamma_T \tag{4.44}$$

where

$$\hat{t} := \arg\max_{1 \leq t \leq T} \hat{\phi}_Y(t/T). \tag{4.45}$$

To prove Theorem 4.1, Lemma 4.5 is applied on the successive iterations of gCPD. In detail, on the event $\mathcal{B}_T$, which has probability larger than $1 - C_3/T$, the following occurs. At the beginning of the algorithm, all conditions of Lemma 4.5 are satisfied. (Note that since no change point has been estimated yet, $\widehat{P} = \widetilde{P} = 0$.) Therefore, the first estimate $\hat{t}^{(1)}$ is at a distance no larger than $TC_2\gamma_T$ from a true change point. At the second iteration, all conditions of Lemma 4.5 are still satisfied, and therefore

$$d(\mathcal{T}^\star | \{\hat{t}^{(1)}, \hat{t}^{(2)}\}) \leq TC_2\gamma_T. \tag{4.46}$$

This goes on, until all change points are detected.

5

Greedy kernel change point detection

Contents

Abstract

This chapter presents gkCPD (Greedy Kernel Change Point Detection), a kernel-based extension of gCPD (see Chapter 4). Thanks to the properties of kernel reproducing Hilbert spaces, gkCPD can detect changes in higher-order moments of probability distributions. We provide several implementation details, that lead to an algorithm with quadratic (in the number of samples) complexity. Nevertheless, gkCPD remains faster than the optimal kernel change point detection algorithm.

1 The rkhs setup for change point detection

1.1 Problem formulation

We consider the segmentation of an $\mathbb{R}^d$-valued signal $\{y_t\}_{t=1}^T$ which, once mapped onto a high-dimensional space, namely a reproducing kernel Hilbert space (RKHS), is piecewise constant with additive noise. The objective is to locate changes in the mean of the mapped signal. Formally, let $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ denote a kernel function and $\mathcal{H}$, the associated RKHS. The related mapping function $\phi : \mathbb{R}^d \to \mathcal{H}$ is implicitly defined by $\phi(y_t) = k(y_t, \cdot) \in$

$\mathcal{H}$ and $\langle \phi(y_s) | \phi(y_t) \rangle_{\mathcal{H}} = k(y_s, y_t)$. The RKHS norm $\|\cdot\|_{\mathcal{H}}$ is also implicitly defined by $\|\phi(y_t)\|_{\mathcal{H}}^2 = k(y_t, y_t)$. We assume that the mapped signal is such that

$$\forall t \in \{t_k^\star + 1, \dots, t_{k+1}^\star\}, \quad \phi(y_t) := \mu_k^\star + \varepsilon_t \tag{5.1}$$

where the $t_k^\star$ ($t_1^\star < \cdots < t_K^\star$) are change point indexes, $t_0^\star := 0$ and $t_{K+1}^\star := T$ are dummy variables, the $\mu_k^\star$ ($k = 0, \dots, K$) are elements of $\mathcal{H}$ and ε_t is a $\mathcal{H}$-valued white noise. Kernel change point detection aims at recovering the unknown set of change points $\mathcal{T}^\star = \{t_1^\star, \dots, t_K^\star\}$. The estimation strategy relies on the minimization of the following kernel least square criterion $V(\cdot)$: for a given set of change points $\mathcal{T}$, we define

$$V(\mathcal{T}) := \sum_{k=0}^{|\mathcal{T}|} \sum_{t=t_k+1}^{t_{k+1}} \|\phi(y_t) - \bar{\mu}_k\|_{\mathcal{H}}^2 \tag{5.2}$$

where $\bar{\mu}_k := \frac{1}{t_{k+1} - t_k} \sum_{t=t_k+1}^{t_{k+1}} \phi(y_t)$, and $t_0 := 0$ and $t_{K+1} := T$ are dummy variables.

1.2 Related work

Within the change point detection framework (Chapter 2), kernel change point detection amounts to setting the cost function to the already introduced c_{kernel}. Such cost functions have emerged because they are non-parametric and model-free, and are able to detect changes in higher-order moments (above the first two) of probability distributions. First introduced in the context of change point detection by [72], it was applied for audio and video segmentation. In this work, like in many works from the literature, the popular Gaussian kernel is used, for its desirable theoretical properties [142] and its numerous successful applications in machine learning [70, 74]. They combine the cost function c_{kernel} with the search method `Opt` (dynamic programming) to estimate the change points, resulting in a complexity of the order of $\mathcal{O}(KT^2)$ where K is the number of changes to estimate and is fixed beforehand. The situation where the parameter K is unknown is tackled in [5] from both a theoretical and methodological point of view. Their procedure has non-asymptotic properties on the convergence of $V(\cdot)$ and has a complexity of the order of $\mathcal{O}(K_{max}T^2)$ where K_{max} is a user-defined upper bound on the number of change points. Consistency results for the estimated change point indexes are provided in a recent work [69]. The observed signal is assumed to be composed of independent random variables with piecewise constant probability distribution. They show that the estimated change point indexes converge with high probability to the true segmentation, when the number of samples grows to infinity, even if the parameter K is unknown. Other authors use c_{kernel} in combination with `Win` (window-based), in more applied contributions Those procedures have been put to use in several areas, notably including audio and video segmentation [75], physiological time series [37, 74] and micro-array segmentation [70].

Notations. It is convenient to introduce, for any $\mathcal{T} = \{t_1, t_2, \dots\}$ and any $\mathcal{H}$-valued signal $\{z_t\}_t$, the orthogonal projection $P_{\mathcal{T}} z$ of z onto the subspace of signals that are constant over the segments delimited by $\mathcal{T}$. As demonstrated in [5], it is given by

$$\forall t \in \{t_k + 1, \dots, t_{k+1}\} \quad (P_{\mathcal{T}} z)_t = \frac{1}{t_{k+1} - t_k} \sum_{s=t_k+1}^{t_{k+1}} z_s. \tag{5.3}$$

Also, in the following, we shall simply refer to the signal $\{\phi(y_t)\}_t$ as $\phi(y)$.

1.3 Contributions of the chapter

In Chapter 4, we have described a trade-off (in terms of complexity) between the exact detection `Opt` and the fast detection `Win`, that outperforms other approximate methods, but is limited to the cost function c_{L_2} and the detection of mean-shifts. We propose to extend our algorithm gCPD to the cost function c_{kernel}. We provide implementation details that allow us to apply the same greedy strategy. The resulting algorithm has desirable complexity properties and is non-parametric and model-free, thanks to the use of a kernel cost function.

2 A kernel version of gCPD

We propose a greedy strategy which is an extension of gCPD with a kernel cost function, with c_{kernel} instead of c_{L_2}. In the following, our algorithm is referred to as gkCPD for "greedy kernel change point detection" and outlined in Algorithm 5.1.

2.1 Reformulation of gCPD with c_{kernel}

The algorithm gkCPD is an iterative procedure that works in the same manner as gCPD. Each iteration consists in two steps: 1) a single change point is detected (greedy detection), 2) its contribution to the original signal is removed with a projection (signal update). The algorithm continues until a stopping criterion is met (which can accommodate K known or unknown, see Section 2.6).

Initialization. At iteration $k = 1$, gkCPD starts by solving the *single* change point detection problem. The first change estimate $\hat{t}^{(1)}$ of gkCPD given by

$$\hat{t}^{(1)} := \underset{t<T}{\arg\min}\ c_{\text{kernel}}(y_{0..t}) + c_{\text{kernel}}(y_{t..T}) \tag{5.4}$$

Then $\phi(y)$ is projected on the subspace of $\mathcal{H}$-valued signals with a single mean-shift located at $\hat{t}^{(1)}$; the resulting residual is denoted $\hat{r}^{(1)}$:

$$\hat{r}^{(1)} := \phi(y) - P_{\widehat{\mathcal{T}}^{(1)}}\phi(y) \tag{5.5}$$

where $\widehat{\mathcal{T}}^{(1)} := \{\hat{t}^{(1)}\}$.

Iteration. After k iterations ($k \geq 1$), the set of already estimated indexes is denoted $\widehat{\mathcal{T}}^{(k-1)} := \{\hat{t}^{(1)}, \dots, \hat{t}^{(k-1)}\}$. The k-th change point estimate $\hat{t}^{(k)}$ is given by

$$\hat{t}^{(k)} := \underset{t<T}{\arg\min}\ c_{\text{kernel}}(\hat{r}^{(k-1)}_{0..t}) + c_{\text{kernel}}(\hat{r}^{(k-1)}_{t..T}). \tag{5.6}$$

where $\hat{r}^{(k-1)}$ is the residual signal from the previous iteration. The index $\hat{t}^{(k)}$ is the solution of the *single* change point detection problem, applied on the $(k-1)$-th residual. The k-th residual $\hat{r}^{(k)} \in \mathcal{H}^T$ is then defined as follows:

$$\hat{r}^{(k)} := \phi(y) - P_{\widehat{\mathcal{T}}^{(k)}}\phi(y) \tag{5.7}$$

where $\widehat{\mathcal{T}}^{(k)} := \{\hat{t}^{(1)}, \dots, \hat{t}^{(k)}\}$ is the set of already estimated indexes, after k iterations. Thus defined, the residual is what remains of the signal $\phi(y)$ after the contributions of the already inferred change points has been "projected" out.

2.2 The gkCPD algorithm

Even though gCPD and gkCPD follow the same principle, the greedy detection (5.6) of gkCPD cannot be performed in the same manner, because the mapping ϕ, and therefore the residual (5.7), are not explicit. To overcome this issue, we express this operation using the inner-products of the signal samples, i.e. $\langle \phi(y_s) | \phi(y_t) \rangle_{\mathcal{H}}$. To that end, introduce the inner-product matrix (or Gram matrix) $G \in \mathbb{R}^{T \times T}$ of the implicit features: $G := [k(y_s, y_t)]_{1 \leq s,t \leq T}$ and the sub-sums of the matrix G:

$$F_{a:b,c:d} := \sum_{s=a+1}^{b} \sum_{t=c+1}^{d} G_{st} \quad (0 \leq a, b, c, d \leq T). \tag{5.8}$$

Assume that $k-1$ iterations have already been performed: the objective is to estimate $\hat{t}^{(k)}$ from the residual $\hat{r}^{(k-1)}$. After simple algebraic manipulations[1], the greedy detection (5.6) can be rewritten as below:

$$\hat{t}^{(k)} = \arg\max_{t} t(T-t) \left\| \bar{r}_{0..t}^{(k-1)} - \bar{r}_{t..T}^{(k-1)} \right\|_{\mathcal{H}}^2 \tag{5.10}$$

where $\bar{r}_{0..t}^{(k-1)}$ and $\bar{r}_{t..T}^{(k-1)}$ are respectively the empirical means of the sub-signals $\{\hat{r}_s^{(k-1)}\}_{s \leq t}$ and $\{\hat{r}_s^{(k-1)}\}_{s > t}$. The quantity to maximize is also known as the *maximum mean discrepancy* (MMD). It is put forth in [70] in a different context to compare the distributions of two sets of samples. According to (5.10), the change point estimate is located where the distributions between the left part of the signal and the right part are the most different. This quantity can be expressed using the inner-products from the residual, yielding

$$\hat{t}^{(k)} = \arg\max_{t < T} \frac{1}{t(T-t)} \sum_{s,u \leq t} \left\langle \hat{r}_s^{(k-1)} \middle| \hat{r}_u^{(k-1)} \right\rangle. \tag{5.11}$$

Using the fact that, by design (5.7), the inner-products $\left\langle r_s^{(k-1)} \middle| r_u^{(k-1)} \right\rangle$ are equal to

$$\langle \phi(y_s) | \phi(y_u) \rangle + \langle f_s | f_u \rangle - \langle f_s | \phi(y_u) \rangle - \langle f_u | \phi(y_s) \rangle \quad \text{where } f := P_{\widehat{\mathcal{T}}^{(k-1)}} \phi(y), \tag{5.12}$$

we are able to derive the following relation:

$$\sum_{s,u=1}^{t} \left\langle \hat{r}_s^{(k-1)} \middle| \hat{r}_u^{(k-1)} \right\rangle = F_{\hat{t}_j:t,\hat{t}_j:t} + \left(\frac{t - \hat{t}_j}{\hat{t}_{j+1} - \hat{t}_j} \right)^2 F_{\hat{t}_j:\hat{t}_{j+1},\hat{t}_j:\hat{t}_{j+1}} - 2 \left(\frac{t - \hat{t}_j}{\hat{t}_{j+1} - \hat{t}_j} \right) F_{\hat{t}_j:t,\hat{t}_j:\hat{t}_{j+1}} \tag{5.13}$$

where $\hat{t}_j$ $(j = 0, \ldots, k)$ is the unique element of $\widehat{\mathcal{T}}^{(k-1)}$ (with $\hat{t}_0 = 0$) such that $\hat{t}_j < t \leq \hat{t}_{j+1}$ and $F_{a:b,c:d}$ is defined in (5.8). By combining (5.11) and (5.13), the greedy estimated $\hat{t}^{(k)}$ can be computed without explicitly calculating the residual signal. The complete algorithm is described in Algorithm 5.1.

[1] More precisely, we use the following relation: for any $\mathcal{H}$-valued signal $\{z_t\}_t$ with T samples and any index $t < T$, we have

$$\sum_{s=1}^{T} \|z_s - \bar{z}\|_{\mathcal{H}}^2 = \left[\sum_{s=1}^{t} \|z_s - \bar{z}_{0..t}\|_{\mathcal{H}}^2 + \sum_{s=t+1}^{T} \|z_s - \bar{z}_{t..T}\|_{\mathcal{H}}^2 \right] + \left[\frac{t(T-t)}{T} \|\bar{z}_{0..t} - \bar{z}_{t..T}\|_{\mathcal{H}}^2 \right] \tag{5.9}$$

where $\bar{z}, \bar{z}_{0..t}, \bar{z}_{t..T}$ are respectively the mean elements of the signals $\{z_s\}_s, \{z_s\}_{s \leq t}, \{z_s\}_{s > t}$.

Algorithm 5.1 gkCPD

Input: inner product matrix G, stopping criterion.
Initialization: $\widehat{\mathcal{T}} \leftarrow \{\}$, $I \leftarrow [\sum_{s\leq i,t\leq j} G_{st}]_{1\leq i,j\leq T}$.
while stopping criterion is not met **do**
    $V \leftarrow [\,]$ ▷ Empty list
    for $t = 1, \dots, T-1$ **do**
        $t_{\text{left}} \leftarrow \max\{s \in \widehat{\mathcal{T}} \cup \{0, T\} | s < t\}$
        $t_{\text{right}} \leftarrow \min\{s \in \widehat{\mathcal{T}} \cup \{0, T\} | s \geq t\}$
        $V[t] \leftarrow F_{t_{\text{left}}:t, t_{\text{left}}:t} + \left(\frac{t-t_{\text{left}}}{t_{\text{right}}-t_{\text{left}}}\right)^2 F_{t_{\text{left}}:t_{\text{right}}, t_{\text{left}}:t_{\text{right}}} - 2\left(\frac{t-t_{\text{left}}}{t_{\text{right}}-t_{\text{left}}}\right) F_{t_{\text{left}}:t, t_{\text{left}}:t_{\text{right}}}$
    end for
    $\hat{t} \leftarrow \arg\max_{t<T} \frac{V[t]}{t(T-t)}$
    $\widehat{\mathcal{T}} \leftarrow \widehat{\mathcal{T}} \cup \{\hat{t}\}$
end while
Output: change point estimates $\widehat{\mathcal{T}}$.

2.3 Complexity analysis

We described how to efficiently compute the Gram matrix of the successive residuals. At first sight, gkCPD has a quadratic complexity, like `Opt` [4]. We show, using the image integral matrix, that gkCPD is faster than `Opt`. Indeed, provided that the matrix

$$I := [\sum_{s\leq i,t\leq j} G_{st}]_{1\leq i,j\leq T} \in \mathbb{R}^{T\times T} \tag{5.14}$$

has been computed, the sub-sums $F_{a:b,c:d}$ are computed in constant time since

$$F_{a:b,c:d} = I_{bd} + I_{ac} - I_{bc} - I_{ad}. \tag{5.15}$$

Therefore, at each iteration, greedy detection (5.11) is performed in linear time. Filling the matrix I is done recursively in quadratic time. Overall complexity of gkCPD is quadratic. The exact change point detection method `Opt` also starts by filling the matrix I and then performs a dynamic programming procedure with a complexity of the order of $\mathcal{O}(KT^2)$ [5, 69, 72]. In gkCPD, this last operation is replaced with an operation with linear complexity. Even though gkCPD and `Opt` have quadratic complexities, gkCPD consequently runs faster.

2.4 Examples of kernels

Our algorithm can be used with various types of data (not just $\mathbb{R}^d$-valued signals), as long as kernel similarity measures are available. In that regard, gkCPD is able adjust to the nature of the signals, which can range from univariate time-series to texts, histograms, etc. Notable examples of kernel functions include [147]:

- The linear kernel $k(x,y) = \langle x|y\rangle$ with $x,y \in \mathbb{R}^d$. Using this kernel is formally equivalent to using the cost function $c_{L_2}(\cdot)$ and therefore gCPD can be used instead of gkCPD.
- The polynomial kernel $k(x,y) = (\langle x|y\rangle + C)^{deg}$ with $x,y \in \mathbb{R}^d$, and C and deg are parameters.

- The Gaussian kernel $k(x,y) = \exp(-\|x-y\|^2/(2\sigma^2))$ with $x,y \in \mathbb{R}^d$.
- The χ^2-kernel $k(x,y) = \exp(-\gamma \sum_i [(x_i - y_i)^2/(x_i + y_i)])$ with $\gamma \in \mathbb{R}$ a parameter. It is often used for histogram data.

3 Conclusion

In this chapter, we described gkCPD, a kernel version of gCPD. Thanks to the properties of reproducing Hilbert spaces, gkCPD detects changes in higher-order moments of probability distributions. As a result of an efficient implementation, this algorithm is faster than its optimal counterpart `Opt`. Numerical experiments are carried out in the next chapter. From a theoretical standpoint, the asymptotic consistency of gkCPD remains an open question. The proof of Chapter 4 cannot be extended because, here, $\phi(y_t)$ is a Hilbert space valued random variable (and not a $\mathbb{R}^d$-valued random variable). A possible approach would be to use concentration inequalities adapted to a non-Gaussian Hilbertian setting, as in [69].

6

Numerical experiments and evaluation

Contents

Abstract

This chapter is dedicated to an empirical comparison of gCPD and gkCPD to standard change point detection algorithms. The evaluation framework (metrics and data sets) is described in Chapter 3. As a result, experiments show that both algorithms display competitive results. In particular, greedy approaches are more accurate than standard sub-optimal methods and faster than optimal methods.

1 Experimental setting

In this chapter we compare the two greedy algorithms gCPD and gkCPD to several standard change point detection methods, on the *MeanShift*, *FreqShift* and *Gait* data sets. Namely, there are eight other methods. Four algorithms rely on the c_{L_2} cost function: `BinSeg` (c_{L_2}) (binary segmentation), `BotUp` (c_{L_2}) (bottom-up segmentation) `Opt` (c_{L_2}) (dynamic programming) and `Win` (c_{L_2}) (window-sliding). Two algorithms rely on the non-parametric c_{rbf} cost function: `Opt` (c_{rbf}) and `Win` (c_{rbf}). Two algorithms rely on parametric cost functions: `Win` (c_{Σ}) and `Win` (c_{AR}). Methods that use the c_{L_2} cost function (including gCPD) can only detect mean-shifts. Methods that use the c_{rbf} cost function (including gkCPD) can detect general distribution changes. The c_{AR} and the c_{Σ} cost functions are respectively sensitive to changes in the autoregressive coefficients of the signal and changes in the mean and covariance matrix of the signal. Optimal methods `Opt`

(c_{L_2}) and `Opt` (c_{rbf}) perform an exhaustive search over the set of signal partitions and return the exact minimum of the sum of costs. Conversely, other methods are sub-optimal and only approximate. The list of all methods is given in Table 6.1. Details about each algorithm and cost function can be found in Chapter 2.
The parameters of the different algorithms are calibrated as follows. In all experiments and for all methods, the number K of change points to detect is assumed to be known. For `BotUp`, the input signal is first divided in 5-sample long sub-signals. For `Win`, the window size is set to 50 samples for the 500-point time series, to 100 for the 2000-point time series and to 100 samples for all time series from the *Gait* data set. For c_{AR}, the autoregressive order is set to $p = 5$. The kernel used is the radial basis function $k(x,y) = \exp(-\gamma\|x-y\|^2)$ where γ is heuristically chosen as the inverse of the empirical median of the pairwise distances, as in [142]. The metrics used to measure the performance of the compared methods include Hausdorff, RandIndex, F1 score (see Chapter 3).

Algorithm	Only mean-shift	More than mean-shift	Applied on
`BinSeg` (c_{L_2})	✓		*MeanShift*, *Gait* (TF)
`BotUp` (c_{L_2})	✓		*MeanShift*, *Gait* (TF)
`gCPD` (c_{L_2})	✓		*MeanShift*, *Gait* (TF)
`gkCPD` (c_{rbf})		✓	*FreqShift* (TF), *Gait* (TF)
`Opt` (c_{L_2})	✓		*MeanShift*, *Gait* (TF)
`Opt` (c_{rbf})		✓	*FreqShift* (TF), *Gait* (TF)
`Win` (c_{Σ})		✓	*FreqShift* (TF), *Gait* (TF)
`Win` (c_{AR})		✓	*FreqShift*, *Gait*
`Win` (c_{L_2})	✓		*MeanShift*, *Gait* (TF)
`Win` (c_{rbf})		✓	*FreqShift* (TF), *Gait* (TF)

Table 6.1: Summary table of compared change point detection methods. “TF” stands for “Time-frequency representation”.

2 Results on the *MeanShift* data set

MeanShift	Metric	gCPD	BinSeg (c_{L_2})	BotUp (c_{L_2})	Win (c_{L_2})	gkCPD	Win (c_{rbf})	Win (c_{Σ})	Win (c_{AR})	Opt (c_{L_2})	Opt (c_{rbf})
Scenario 1	Hausdorff	0.32 (±0.58)	0.23 (±0.51)	2.13 (±0.80)	0.43 (±0.67)	0.28 (±0.58)	0.30 (±0.56)	65.00 (±38.39)	1.60 (±2.01)	0.08 (±0.27)	0.08 (±0.27)
	RandIndex	1.00 (±0.00)	1.00 (±0.00)	0.99 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	0.90 (±0.05)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)
	F1 score	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	0.54 (±0.20)	0.99 (±0.03)	1.00 (±0.00)	1.00 (±0.00)
Scenario 2	Hausdorff	5.55 (±5.06)	7.18 (±10.48)	7.96 (±4.74)	29.62 (±35.95)	15.97 (±26.68)	41.06 (±40.68)	80.45 (±29.71)	81.92 (±37.48)	4.29 (±3.61)	4.51 (±4.16)
	RandIndex	0.99 (±0.01)	0.98 (±0.01)	0.98 (±0.01)	0.96 (±0.04)	0.98 (±0.03)	0.94 (±0.04)	0.85 (±0.04)	0.87 (±0.06)	0.99 (±0.01)	0.99 (±0.01)
	F1 score	0.95 (±0.12)	0.94 (±0.13)	0.91 (±0.15)	0.85 (±0.16)	0.91 (±0.16)	0.82 (±0.17)	0.31 (±0.21)	0.47 (±0.23)	0.97 (±0.10)	0.96 (±0.10)
Scenario 3	Hausdorff	0.28 (±0.53)	0.36 (±0.67)	2.17 (±0.63)	1.42 (±0.49)	0.31 (±0.54)	1.38 (±0.49)	10.55 (±35.68)	1.78 (±0.97)	0.13 (±0.34)	1.69 (±0.48)
	RandIndex	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.01)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)
	F1 score	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	0.96 (±0.09)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)
Scenario 4	Hausdorff	4.63 (±5.95)	5.35 (±6.71)	7.68 (±4.86)	10.34 (±27.14)	5.80 (±7.14)	16.28 (±55.17)	377.55 (±130.63)	225.09 (±182.39)	3.14 (±2.60)	4.11 (±2.77)
	RandIndex	1.00 (±0.00)	1.00 (±0.00)	0.99 (±0.00)	0.99 (±0.01)	1.00 (±0.00)	0.99 (±0.01)	0.84 (±0.06)	0.93 (±0.05)	1.00 (±0.00)	1.00 (±0.00)
	F1 score	0.99 (±0.03)	0.99 (±0.05)	1.00 (±0.02)	0.99 (±0.05)	0.99 (±0.05)	0.99 (±0.05)	0.37 (±0.23)	0.76 (±0.19)	1.00 (±0.00)	1.00 (±0.00)

Table 6.2: Means and standard deviations on the *MeanShift* data set are shown. Hausdorff is expressed in number of samples. The margin of F1 score is $M = 10$ samples for Scenario 1 and 2, and $M = 20$ samples for Scenario 3 and 4.

In this section, all ten detection methods are compared on the *MeanShift* data set, described in Section 2.2.2 on page 83. Several observations can be made from the results that are reported in Table 6.2.

- **Compared to the other algorithms, window-based methods are less robust to noise.** Overall, those methods have the most important decrease in performance when going from Scenario 1 to Scenario 2, and from Scenario 3 to Scenario 4, i.e. from $\sigma = 1$ to $\sigma = 3$. For instance, on Scenario 1 ($\sigma = 1$), `Win` (c_{L_2}) is close to optimal according to all metrics, but on Scenario 2 ($\sigma = 3$), it is the least accurate method among the the ones that use the c_{L_2} cost function (HAUSDORFF is at 29.62 samples on average, while the second worst is at 7.96). The same phenomenon is observed for `Win` (c_{AR}) and `Win` (c_{rbf}) when going from Scenario 3 to Scenario 4. This can be explained by the fact that only a handful of samples are used to detect a change point. Cost functions which rely on the estimation of several statistical quantities are even more penalized by the lack of samples. This is the reason why `Win` (c_{Σ}) has poor performances, compared to other methods: it estimates the empirical mean and empirical variance/covariance matrix of 20-dimensional signals using 25 or 50 samples (length of half a window). To a smaller extent, the same applies to `Win` (c_{AR}) which has to estimate $p = 5$ coefficients. In addition, the signals do not follow a (vector) autoregressive model. One solution to mitigate the lack of samples is to increase window length. However, the number of samples cannot grow beyond a certain limit because window-based methods are valid as long as there is at most one change point within each window. Calibration of the window length is a common issue [75]. To conclude, the search method `Win` is better adapted to detect *rare* change points, with cost functions that rely on the estimation of only a few parameters.

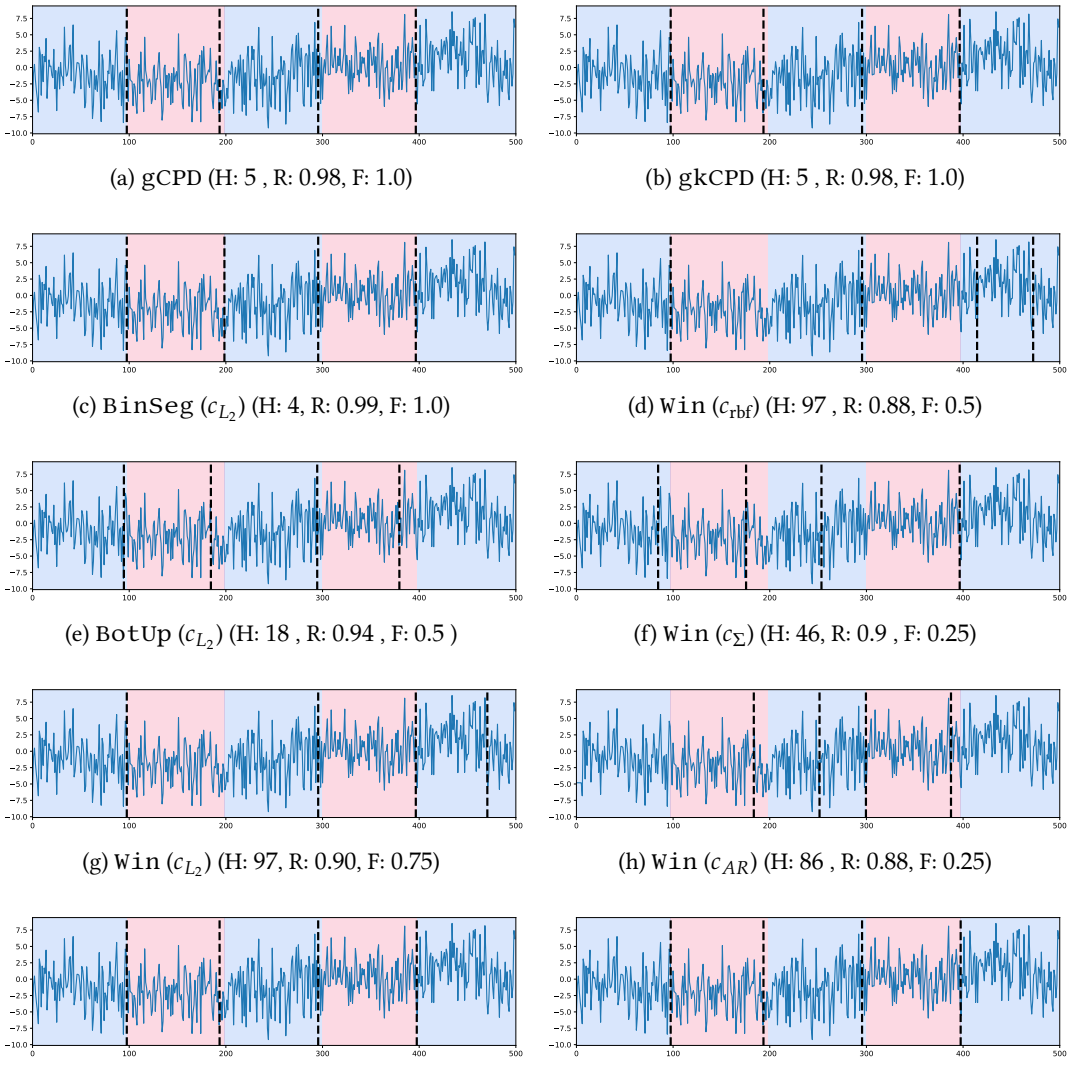
- **Methods that use the c_{L_2} cost function perform well on this data set.** All approximate methods with the c_{L_2} cost function (except `Win`, see previous observation) have similar performances, especially on Scenario 1 and Scenario 3, where $\sigma = 1$. According to all metrics, they are close to the optimal algorithm `Opt` (c_{L_2}). A closer look reveals that `gCPD` is the most accurate (often by a thin margin) and `BotUp` (c_{L_2}) has its HAUSDORFF value always above two samples (a consequence of the initial segmentation in 5-sample long sub-signals). Such performances are expected because the signals of this data set follow the exact model (piecewise constant with additive Gaussian white noise) for which those methods were introduced.

- **Using a kernel does bring about a significant gain.** For the `Opt` algorithms, both c_{L_2} and c_{rbf} have almost equal scores. The kernel-based alternative `gkCPD` does not improve the segmentation performance of its counterpart `gCPD`. On Scenario 2 (the most difficult), `gkCPD` is less precise than `gCPD` according to HAUSDORFF and F1 SCORE. This can be explained by the fact that c_{rbf} consider a general class of change points while c_{L_2} focuses on changes in the mean of Gaussian variables, exactly the type of changes present in *MeanShift*. When using the optimal search method `Opt`, both are equivalently precise, but with the approximate greedy method, and on the most difficult scenario, a difference can be observed.

- **Segmentation example.** These observations are illustrated on a segmentation example displayed on Figure 6.1. The most accurate segmentation algorithms on this

example are gCPD, gkCPD, `BinSeg` (c_{L_2}), `Opt` (c_{L_2}) and `Opt` (c_{rbf}): all change points are recovered and worst error is 5 samples. Here, `BotUp` (c_{L_2}) is less accurate, which is also true on average according to results in Table 6.2. Window-based methods are visibly less precise.

To sum up, window-based methods perform relatively worse than other methods, because of the limited number of samples they use. Algorithms with the c_{L_2} cost function perform well on this data set, to the point that kernel methods are not necessary.



(a) gCPD (H: 5 , R: 0.98, F: 1.0)

(b) gkCPD (H: 5 , R: 0.98, F: 1.0)

(c) `BinSeg` (c_{L_2}) (H: 4, R: 0.99, F: 1.0)

(d) `Win` (c_{rbf}) (H: 97 , R: 0.88, F: 0.5)

(e) `BotUp` (c_{L_2}) (H: 18 , R: 0.94 , F: 0.5)

(f) `Win` (c_{Σ}) (H: 46, R: 0.9 , F: 0.25)

(g) `Win` (c_{L_2}) (H: 97, R: 0.90, F: 0.75)

(h) `Win` (c_{AR}) (H: 86 , R: 0.88, F: 0.25)

(i) `Opt` (c_{L_2}) (H: 5 , R: 0.99, F: 1.0)

(j) `Opt` (c_{rbf}) (H: 5, R: 0.99 , F: 1.0)

Figure 6.1: Segmentation examples on a signal from *MeanShift* (Scenario 2). H, R and F respectively denote HAUSDORFF, RANDINDEX and F1 SCORE. The alternating coloured areas denote the true segmentation. Dashed lines mark estimated change points. Only one dimension of the signal is displayed. The complete signal is displayed on Figure 3.7 on page 85.

3 Results on the *FreqShift* data set

FreqShift	Metric	gCPD	BinSeg (c_{L_2})	BotUp (c_{L_2})	Win (c_{L_2})	gkCPD	Win (c_{rbf})	Win (c_{Σ})	Win (c_{AR})	Opt (c_{L_2})	Opt (c_{rbf})
−5 dB	Hausdorff	57.96 (±64.98)	79.84 (±95.81)	53.08 (±38.93)	368.34 (±137.01)	48.28 (±57.06)	374.48 (±132.76)	367.48 (±160.33)	405.47 (±186.93)	34.12 (±51.28)	27.98 (±26.66)
	RandIndex	0.97 (±0.02)	0.96 (±0.04)	0.96 (±0.02)	0.84 (±0.07)	0.97 (±0.02)	0.84 (±0.06)	0.82 (±0.07)	0.79 (±0.07)	0.98 (±0.02)	0.98 (±0.01)
	F1 score	0.68 (±0.21)	0.62 (±0.23)	0.58 (±0.24)	0.35 (±0.23)	0.71 (±0.21)	0.32 (±0.22)	0.17 (±0.17)	0.10 (±0.15)	0.81 (±0.19)	0.82 (±0.18)
−1 dB	Hausdorff	20.86 (±12.70)	23.86 (±17.75)	25.17 (±12.08)	148.59 (±163.52)	17.80 (±7.31)	189.09 (±166.78)	348.06 (±134.19)	407.57 (±191.55)	13.59 (±6.83)	12.76 (±4.91)
	RandIndex	0.98 (±0.01)	0.98 (±0.01)	0.98 (±0.01)	0.95 (±0.04)	0.98 (±0.01)	0.94 (±0.05)	0.85 (±0.06)	0.80 (±0.07)	0.99 (±0.00)	0.99 (±0.00)
	F1 score	0.85 (±0.17)	0.84 (±0.18)	0.74 (±0.22)	0.77 (±0.19)	0.90 (±0.13)	0.71 (±0.19)	0.21 (±0.20)	0.14 (±0.16)	0.97 (±0.09)	0.97 (±0.08)
0 dB	Hausdorff	19.73 (±8.90)	21.98 (±9.95)	22.15 (±8.02)	97.12 (±148.83)	18.67 (±8.38)	110.19 (±152.74)	355.42 (±164.17)	367.29 (±142.69)	11.88 (±4.73)	11.45 (±4.13)
	RandIndex	0.98 (±0.01)	0.98 (±0.01)	0.98 (±0.01)	0.97 (±0.04)	0.98 (±0.01)	0.96 (±0.04)	0.84 (±0.08)	0.81 (±0.07)	0.99 (±0.00)	0.99 (±0.00)
	F1 score	0.87 (±0.17)	0.84 (±0.18)	0.79 (±0.21)	0.86 (±0.16)	0.90 (±0.15)	0.83 (±0.18)	0.25 (±0.19)	0.18 (±0.19)	0.99 (±0.05)	0.99 (±0.04)
2 dB	Hausdorff	18.36 (±6.08)	20.56 (±5.58)	21.56 (±7.09)	17.84 (±39.04)	16.10 (±7.56)	20.68 (±39.89)	345.68 (±179.53)	347.07 (±152.22)	9.68 (±4.12)	9.64 (±3.66)
	RandIndex	0.98 (±0.00)	0.98 (±0.00)	0.98 (±0.01)	0.99 (±0.01)	0.99 (±0.01)	0.99 (±0.01)	0.86 (±0.07)	0.83 (±0.07)	0.99 (±0.00)	0.99 (±0.00)
	F1 score	0.90 (±0.14)	0.83 (±0.17)	0.79 (±0.23)	0.95 (±0.12)	0.93 (±0.13)	0.92 (±0.14)	0.22 (±0.21)	0.25 (±0.20)	0.99 (±0.04)	0.99 (±0.03)

Table 6.3: Performance (means and standard deviations) on the *FreqShift* data set. Margin for F1 score is 20 samples. Hausdorff is in number of samples.

In this section, all ten detection methods are compared on the *FreqShift* data set, described in Section 2.2.2 on page 86. Except `Win` (c_{AR}), all algorithms are applied on the time-frequency representation of the signals. Several observations can be made from the results that are reported in Table 6.3.

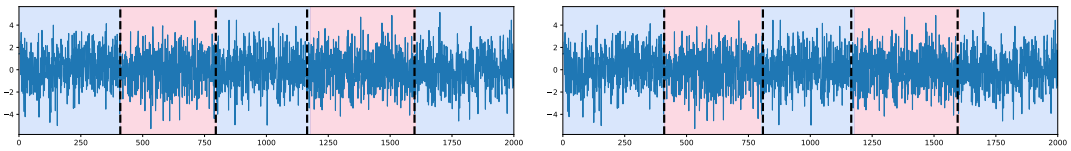
- **Segmentation is more difficult on this data set.** Compared to *MeanShift*, which is an "ideal" data set, all algorithms are less accurate, even though the normalized mean-shift amplitudes (see Table 3.1 on page 87) are larger on the *FreqShift* data set. This is evidenced for instance by the Hausdorff values of both `Opt` (c_{L_2}) and `Opt` (c_{rbf}) which are around ten samples at best (SNR $= 0{,}2$ dB), while they are around four samples at worst on *MeanShift*. This is explained by several factors. The number of dimensions is greater (50 for the time-frequency representation, 20 for *MeanShift*) and few dimensions contain a mean-shift (see the spectrogram displayed on Figure 3.8 on page 86). The remaining dimensions only contain noise. Also, the coefficients of the the time-frequency representation follow a chi-squared distribution rather than a Gaussian distribution [127]. Lastly, the trade-off between time resolution and frequency resolution of such representations limits the detection accuracy of the change points.

- **Window-based methods are less robust to noise.** The same phenomenon as in the *MeanShift* experiment is observed. The most illustrative example is `Win` (c_{Σ}) which performs almost equally bad for all noise levels. Interestingly, `Win` (c_{L_2}) and `Win` (c_{rbf}) have relatively good scores when SNR$=2$ dB. In this setting, window-based have a cut-off SNR level, above which they perform relatively well and under which they perform poorly. Their estimations remain unstable, as indicated by the high variance of Hausdorff, compared to other methods with the same performance.

- **The algorithm `gkCPD` is generally more accurate.** Using the kernel-based `gkCPD` provides a more accurate segmentation, compared to the c_{L_2}-based methods. For SNR $= -5$ dB, the second best algorithm is `BotUp`, but when SNR increases to -1 dB, it does not improve as much as the other methods and `gCPD` is then the second best performing algorithm. This also holds for SNR $= 0$ dB. When SNR is highest, `Win` (c_{L_2}) is the closest method to `gkCPD`. This observation can be explained by the fact that `gkCPD` relies on a kernel and noise in *FreqShift* signals (time-frequency representation) is not Gaussian but distributed according a chi-squared.

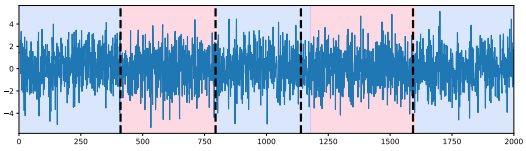
- **Segmentation example.** These observations are illustrated on a segmentation example displayed on Figure 6.1. The best methods on this example are `gCPD` and `gkCPD`, which outperform on this particular signal `Opt` (c_{L_2}) and `Opt` (c_{rbf}). `BinSeg` (c_{L_2}), `Win` (c_{rbf}) and `BotUp` (c_{L_2}) have less accurate: they all miss one change point by 30-40 samples (which yields a F1 score of 0.75). `Win` (c_{L_2}) has a similar Hausdorff value but misses two change points by 30 samples, which yields a F1 score of 0.5.

To sum up, as in *MeanShift*, window-based methods perform quite worse than other methods, except when the SNR is high enough (at 2 dB in this experiment). Contrary to the *MeanShift* experiment, `gkCPD` improve the segmentation accuracy of algorithms that use c_{L_2}. Using a kernel is more appropriate when the signals differ, even slightly, from the "ideal" model of a piecewise constant signal with additive Gaussian white noise.

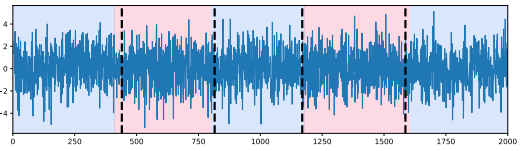


(a) gCPD (H: 14 , R: 0.99, F: 1.0)

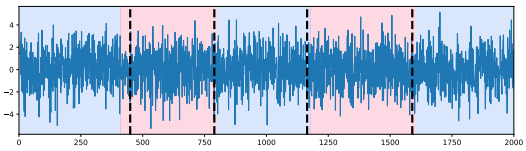
(b) gkCPD (H: 13 , R: 0.99, F: 1.0)



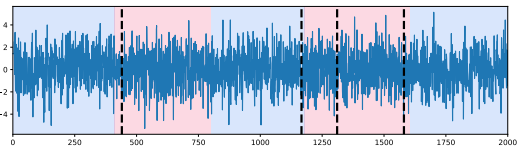
(c) `BinSeg` (c_{L_2}) (H: 39 , R: 0.98, F: 0.75)



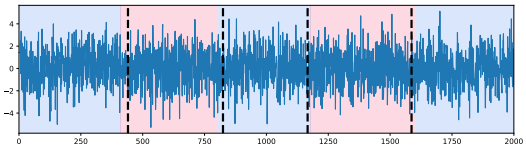
(d) `Win` (c_{rbf}) (H: 31 , R: 0.97, F: 0.75)



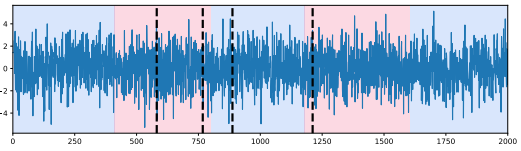
(e) `BotUp` (c_{L_2}) (H: 40 , R: 0.97, F: 0.75)



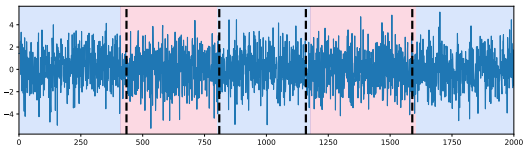
(f) `Win` (c_{Σ}) (H: 359 , R: 0.89, F: 0.25)



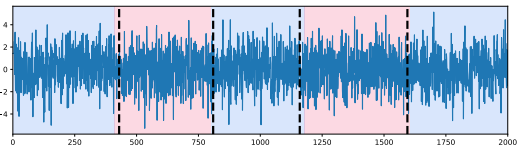
(g) `Win` (c_{L_2}) (H: 31 , R: 0.97, F: 0.5)



(h) `Win` (c_{AR}) (H: 393 , R: 0.84, F: 0.0)



(i) `Opt` (c_{L_2}) (H: 25 , R: 0.97, F: 0.75)



(j) `Opt` (c_{rbf}) (H: 20 , R: 0.98, F: 0.75)

Figure 6.2: Segmentation example on a signal from *FreqShift* (SNR $= -1$ dB). H, R and F respectively denote HAUSDORFF, RANDINDEX and F1 SCORE. The alternating coloured areas denote the true segmentation. Dashed lines mark estimated change points. The time-frequency representation of the signal is displayed on Figure 3.8 on page 86.

4 Results on the *Gait* data set

In this section, all ten detection methods are compared on the *Gait* data set, described in Section 2.2.1 on page 81. Global results are reported in Table 6.4 and the accuracy by change point type is reported in Table 6.5.

4.1 Global results

Metric	gCPD	BinSeg (c_{L_2})	BotUp (c_{L_2})	Win (c_{L_2})	Opt (c_{L_2})
Hausdorff	1.34 (±0.58)	4.44 (±3.38)	3.07 (±3.18)	2.92 (±3.21)	1.80 (±2.35)
RandIndex	0.90 (±0.03)	0.92 (±0.02)	0.92 (±0.03)	0.91 (±0.05)	0.92 (±0.02)
F1 score	0.73 (±0.20)	0.79 (±0.13)	0.78 (±0.16)	0.81 (±0.17)	0.85 (±0.13)

(a) Detection with the c_{L_2} cost function.

Metric	gkCPD	Win (c_{rbf})	Win (c_{Σ})	Win (c_{AR})	Opt (c_{rbf})
Hausdorff	1.13 (±0.67)	3.31 (±3.44)	6.06 (±3.87)	6.35 (±2.23)	1.29 (±1.06)
RandIndex	0.91 (±0.02)	0.91 (±0.05)	0.78 (±0.12)	0.81 (±0.04)	0.91 (±0.02)
F1 score	0.85 (±0.15)	0.80 (±0.18)	0.54 (±0.18)	0.51 (±0.17)	0.83 (±0.15)

(b) Detection with the c_{rbf}, c_{Σ} and c_{AR} cost functions.

Table 6.4: Performance (means and standard deviations) on the *Gait* data set. Margin for F1 score is one second. Hausdorff is in second.

Several observations can be made from the results that are reported in Table 6.4. Interpretation is here more complex than it was for the synthetic data sets *MeanShift* and *FreqShift*. Depending on the metric, the ranking of algorithms is not always the same.

- Hausdorff **measures the worst estimation error.** A small Hausdorff values indicates that all change points are correctly estimated, and a large Hausdorff indicates that *at least one* change point is incorrectly estimated. By looking at this metric, we observe the following.
  - According to Hausdorff, the two best approximate methods are the greedy ones, gCPD and gkCPD. Each has even a better score than its optimal counterpart, Opt (c_{L_2}) or Opt (c_{rbf}). Here, optimal segmentation is not necessarily best when the signals do not follow the model assumed by the cost function.
  - The less accurate methods on this data set are Win (c_{Σ}) and Win (c_{AR}). As observed in previous experiments, those cost functions have to estimate many parameters, which can lead to poor performance when noise level is large and the data do not follow the assumed model. Win (c_{L_2}) fares better than Win (c_{rbf}), for similar reasons.
- RandIndex **measures the proportion of agreement between two segmentations.** A RandIndex value of $0.x$ indicates that it is x% likely that a pair of samples are in the same regime or in different regimes according to both segmentations. Contrary to Hausdorff, not all change points are treated the same: errors on short regimes weight less than errors on long regimes in the computation of this metric. By looking at this metric, we observe the following.
  - All method, except Win (c_{Σ}) and Win (c_{AR}) have a RandIndex around 0.9. Tree-based methods, BinSeg (c_{L_2}) and BotUp (c_{L_2}), are slightly better (0.92) and window-based methods, Win (c_{L_2}) and Win (c_{rbf}), have a slightly higher variance. This indicates that long regimes (i.e. "Stand" and the two "Walk" phases) are better recovered than short regimes ("Stop" phases).

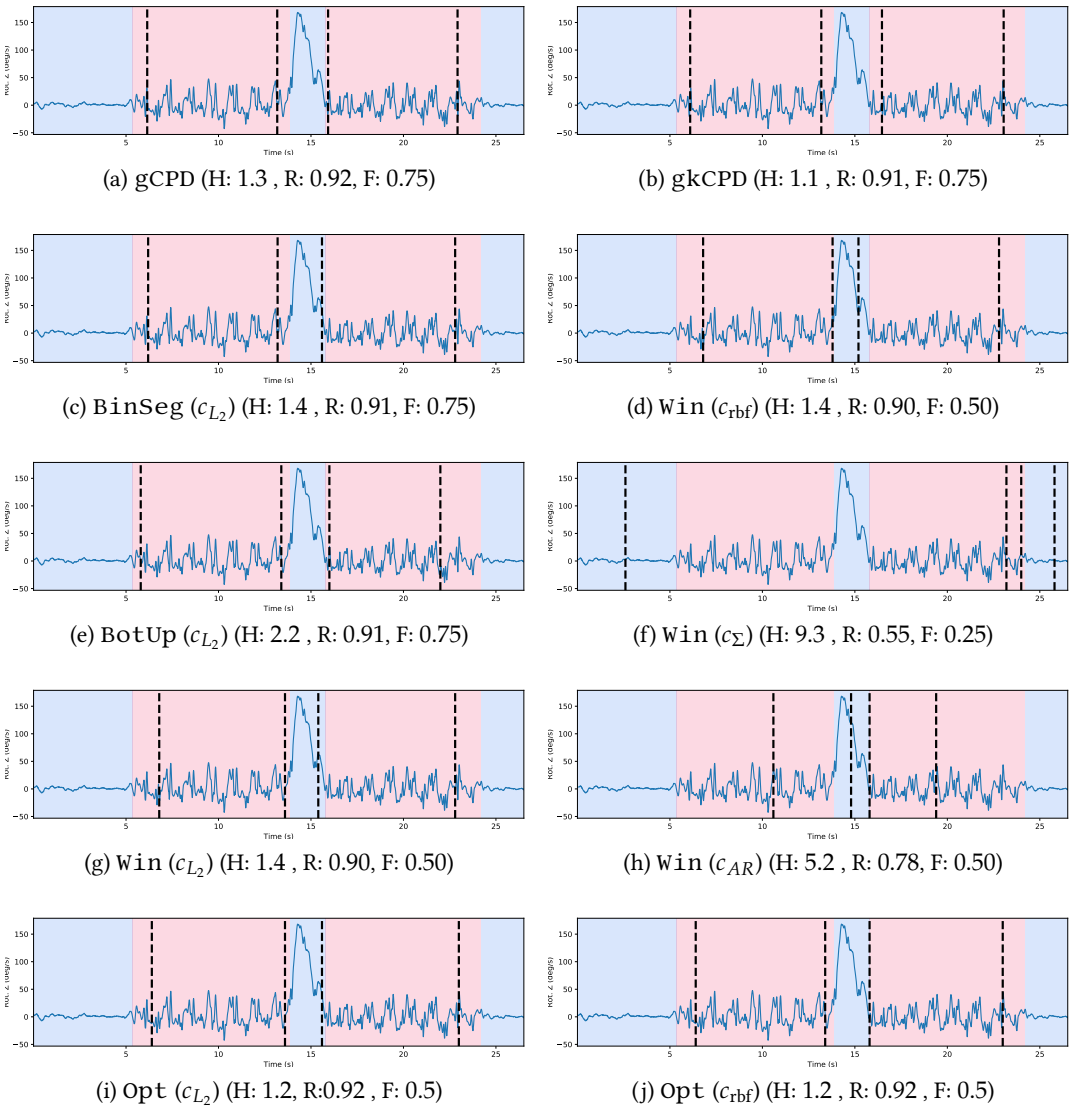
  - Again, the less accurate methods on this data set are `Win` (c_Σ) and `Win` (c_{AR}) with a RandIndex around 0.80.

- F1 score **measures the precision and recall of the estimation.** Missing a "true" change point and placing several estimations around one "true" change point both deteriorate the F1 score. By looking at this metric, we observe the following.

  - Certain methods that have a Hausdorff value well above one second (the margin of F1 score) still have good F1 score values (around 0.80), for instance `Win` (c_{L_2}) and `Win` (c_{rbf}). This indicates that those algorithms tend to miss a change point by a large margin (resulting in a high Hausdorff) but the remaining change points are well estimated.

  - The least accurate methods are `Win` (c_Σ) and `Win` (c_{AR}). They both seem to miss two out of the four change points (F1 score around 0.5). The most accurate method is `gkCPD` (F1 score of 0.85).

These observations are illustrated on a segmentation example displayed on Figure 6.3. The best method on this signal is `gkCPD`, followed by `Opt`(c_{L_2}) and `Opt` (c_{rbf}). A common behaviour, shared by methods with a RandIndex above 0.90 (for instance `gCPD`) is to include in the "Stand" phase the first footstep of the "Walk" phase, and to include in the "Stop" phase the last step of the "Walk" phase. This can be explained by the fact that either the first or the last footstep has a smaller amplitude than the others.
To sum up, the segmentation results on the *Gait* data set are more complex to interpret. When looking at each metric individually, certain trends are observed. Long regimes are well recovered, resulting in a RandIndex above 0.90 for several algorithms. Parametric window-based methods are not adapted to the signals. Other algorithms miss less than one change point on average according to the F1 score.



(a) gCPD (H: 1.3 , R: 0.92, F: 0.75)

(b) gkCPD (H: 1.1 , R: 0.91, F: 0.75)

(c) `BinSeg` (c_{L_2}) (H: 1.4 , R: 0.91, F: 0.75)

(d) `Win` (c_{rbf}) (H: 1.4 , R: 0.90, F: 0.50)

(e) `BotUp` (c_{L_2}) (H: 2.2 , R: 0.91, F: 0.75)

(f) `Win` (c_{Σ}) (H: 9.3 , R: 0.55, F: 0.25)

(g) `Win` (c_{L_2}) (H: 1.4 , R: 0.90, F: 0.50)

(h) `Win` (c_{AR}) (H: 5.2 , R: 0.78, F: 0.50)

(i) `Opt` (c_{L_2}) (H: 1.2, R:0.92 , F: 0.5)

(j) `Opt` (c_{rbf}) (H: 1.2 , R: 0.92 , F: 0.5)

Figure 6.3: Segmentation example on a signal from *Gait*. H, R and F respectively denote HAUSDORFF (in seconds), RANDINDEX and F1 SCORE. The alternating coloured areas denote the true segmentation. Dashed lines mark estimated change points. The acceleration on the (0z) axis is shown.

4.2 Results by change point type

Gait	gCPD	BinSeg (c_{L_2})	BotUp (c_{L_2})	Win (c_{L_2})	Opt (c_{L_2})
Stand/Walk	0.37 (±0.32)	0.53 (±0.91)	0.60 (±1.00)	2.00 (±2.81)	0.60 (±0.92)
Walk/Turnaround	0.88 (±0.69)	0.57 (±0.64)	0.41 (±0.74)	0.94 (±1.58)	0.38 (±0.62)
Turnaround/Walk	0.99 (±0.95)	0.69 (±1.00)	0.51 (±0.59)	1.28 (±2.11)	0.38 (±0.43)
Walk/Stop	1.01 (±0.52)	4.32 (±3.41)	2.89 (±3.19)	1.42 (±2.39)	1.69 (±2.24)

(a) Detection with the c_{L_2} cost function.

Gait	gkCPD	Win (c_{rbf})	Win (c_{Σ})	Win (c_{AR})	Opt (c_{rbf})
Stand/Walk	0.48 (±0.33)	2.20 (±2.95)	2.45 (±1.92)	5.26 (±2.37)	0.52 (±0.32)
Walk/Turnaround	0.65 (±0.64)	0.93 (±1.57)	6.13 (±2.94)	1.04 (±1.27)	0.62 (±0.81)
Turnaround/Walk	0.63 (±0.78)	1.29 (±2.07)	4.85 (±3.37)	1.85 (±2.07)	0.49 (±0.43)
Walk/Stop	0.98 (±0.72)	1.69 (±2.76)	2.87 (±4.72)	3.42 (±3.09)	1.22 (±1.09)

(b) Detection with the c_{rbf}, c_{Σ} and c_{AR} cost functions.

Table 6.5: Average temporal distance of a predicted change point to an annotated change point (in seconds). Means and standard deviations on the *Gait* data set are shown.

In our context, it is important to notice that all change points are not equivalent. They limit phases of different natures. We provide the details of the segmentation results by change point type in Table 6.5. Several observations can be made.

- The first change point is the best detected by six out of the ten algorithms, namely gCPD, BinSeg (c_{L_2}), BotUp (c_{L_2}), Opt (c_{L_2}), gkCPD and Opt (c_{rbf}). The average temporal error for those algorithms is around 0.50 second. This can be explained by the fact that this change point separates two long regimes which are visibly very different ("Stand" is somewhat flat, and "Walk" has a large amplitude). Interestingly, window-based methods are not as precise on this particular change point. Since the search for a change is limited to a small region (the window), they cannot take advantage of the length of the regimes.

- Even though it is of the same type as the first one, estimation of the last change point is generally less accurate. (Both change points separate a "Walk" phase and a rest phase ("Stand" or "Stop").) Two reasons can justify this observation. The last change point corresponds to a smaller mean-shift amplitude (in the time-frequency representation). Also, it is located at the edge of the signal, which can influence certain methods such as BinSeg (c_{L_2}) and BinSeg (c_{L_2}).

- For algorithms that use the c_{L_2} and c_{rbf} cost functions, the error on the last change point is what drives the Hausdorff score (see Table 6.4): it is the worst estimation error. It is interesting to note that BinSeg (c_{L_2}) and BinSeg (c_{L_2}) have errors comparable to gCPD and gkCPD on the first three changes but are three to four times less precise on the last change point. This explains why those four methods have similar RandIndex values (error is made on the short regime "Stop") but different Hausdorff values.

- The "Turnaround" is well detected by four approximate algorithm (estimation error below one second), namely gCPD, BinSeg (c_{L_2}), BotUp (c_{L_2}), and gkCPD The best one for this phase is BotUp (c_{L_2}), with an error close to the one of Opt (c_{L_2}).

After a closer look on the segmentations, we observe that gkCPD and gCPD tend to include in the "Turnaround" phase the last footstep of the previous "Walk" phase and the first footstep of the previous "Walk" phase. This behaviour is displayed on the segmentation example on Figure 6.3. This is understandable because the footsteps at the beginning or end of each regime can be different from the footsteps in the middle of the regime (for instance, of smaller amplitude because the subject is accelerating or slowing down).

- The two algorithms `Win` (c_{L_2}) and `Win` (c_{rbf}) have F1 SCORE around 0.80 (Table 6.4) but on three out of four change points, the average temporal error is well above the margin of one second. This indicates that on average, three change points are correctly estimated (thus the F1 SCORE above 0.75), but the last one is greatly misplaced. Also, those algorithms make an error indiscriminately on either the first ("Stand/Walk"), the third ("Turnaround/Walk") or the fourth ("Walk/Stop"), as evidenced by the high temporal distances. Conversely, `BotUp` (c_{L_2}) is more likely to make an error on the last one only.

To sum up, observing the error by change point type allows us to better understand the behaviour of segmentation methods. Window-based methods are confirmed not to be as precise as the other methods on this data set. Tree-based methods algorithms, `BotUp` (c_{L_2}) and `BinSeg` (c_{L_2}) are precise on the first three change points but misplace the last one by a large margin. The greedy procedures gCPD and gkCPD are able to locate all change points with an error of less than one second. In particular, gkCPD is relatively more accurate on average.

5 Discussion

5.1 Execution time comparison

Data set	gCPD	`BinSeg` (c_{L_2})	`BotUp` (c_{L_2})	`Win` (c_{L_2})	`Opt` (c_{L_2})
MeanShift ($T = 500$)	0.2	17.6	3.0	6.6	7 min
MeanShift ($T = 2000$)	0.6	129.4	22.7	23.8	42 min
FreqShift ($T = 2000$)	0.8	214.3	20.4	23.5	2 h
Gait data set	1.0	235.0	32.6	26	4 h

(a) Detection with the c_{L_2} cost function.

Data set	gkCPD	`Win` (c_{rbf})	`Win` (c_{Σ})	`Win` (c_{AR})	`Opt` (c_{rbf})
MeanShift ($T = 500$)	6.1	2.8	40.0	22.2	6 min
MeanShift ($T = 2000$)	89.6	28.6	193.0	96.1	11 h
FreqShift ($T = 2000$)	74.5	27.6	146.3	61.5	$>$ 1 day
Gait data set	94.2	22.13	161.7	65.7	$>$ 2 days

(b) Detection with the c_{rbf}, c_{Σ} and c_{AR} cost functions.

Table 6.6: Average runtime for each algorithm to process 100 signals. All times refer to a Python implementation on a Linux computer with 4 processors running at 2.80 GHz. All times are in seconds, unless specified otherwise.

Table 6.6 presents average execution times (for 100 signals) of the different methods. Visibly, differences in execution time increase as the number of samples and the number

of dimensions grows. Even if gCPD, `BinSeg` (c_{L_2}) and `BotUp` (c_{L_2}) all have a linear (in the number of samples) computational complexity, gCPD's implementation ease allows for an efficient execution. Specifically, operations described in Algorithm 4.1 are naturally "vectorized". In languages like Python and Matlab, such operations are more cost-effective than the explicit looping instructions needed in tree-based methods. As a comparison, the signal acquisition system takes around 50 seconds to record one signal, while gCPD takes 1 second to process 100 signals, and gkCPD takes less than 100 seconds. `Opt` (c_{L_2}) is far slower, as expected. All `Win` methods do not take the same amount of time, because, the computation of the cost function can be lead to significant differences in complexity. For instance c_{L_2} only require to calculate the empirical mean, while the c_{rbf} requires the computation of all pairwise kernel products. Note that the implementation of `Win` (c_{L_2}), in the `ruptures` package (see Chapter 9) is not optimized for this particular algorithm, but rather focus on the modularity of the package. One advantage `Win` (c_{L_2}) (and other window-based methods) has over gCPD is the fact that it only processes a portion of the signal at a time. When memory is an issue or when faced with a continuous stream, `Win` (c_{L_2}) is the only appropriate method, as other methods are performed on the whole signal.

As for gkCPD, it is more computationally intensive than its counterpart gCPD, but remains faster than `Opt` (c_{rbf}).

5.2 Estimation of the number of change points with gCPD

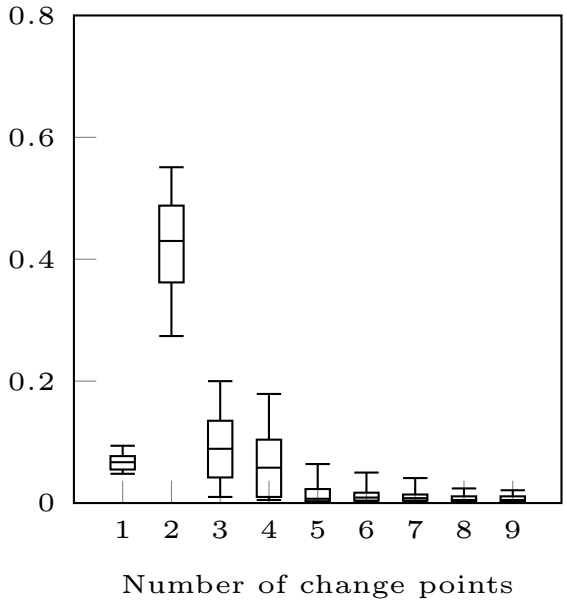


Figure 6.4: Gains (6.1) of gCPD on the *Gait* data set. The $[0.1, 0.25, 0.5, 0.75, 0.9]$-percentiles are shown.

In our experiments, the number of change point is known beforehand. However, in certain applications, such information might not be known. We show in the following that our greedy strategy can be combined with a simple model selection procedure to accommodate situations where the number of changes is unknown. To that end, we start off by looking at

Number of change points k	BIC(k)
0	1334.67 ($\pm$501.13)
1	1270.75 ($\pm$436.21)
2	784.35 ($\pm$342.97)
3	687.51 ($\pm$256.30)
4	**639.95** ($\pm$**232.50**)
5	660.26 ($\pm$222.77)
6	682.14 ($\pm$206.80)
7	709.80 ($\pm$200.73)
8	742.82 ($\pm$194.83)
9	776.26 ($\pm$190.08)

Table 6.7: Constrained costs of the sequential estimates of gCPD. Mean and standard deviation are displayed.

the evolution of gCPD's residuals. By design, the sequence of the residuals $\widehat{R}^{(k)}$ produced at each each step of gCPD is strictly decreasing in norm. The gain of the k^{th} step is

$$\frac{\left\|\widehat{R}^{(k-1)}\right\|^2 - \left\|\widehat{R}^{(k)}\right\|^2}{\left\|\widehat{R}^{(0)}\right\|^2} \tag{6.1}$$

where $\left\|\widehat{R}^{(0)}\right\|^2 = \|Y\|^2$ acts as a normalization constant. Repartition of the gain values are reported on Figure 6.4. The first four steps of gCPD have greater gains than the following ones. Gains after step $k = 5$ bring little to no gain; they are more concentrated around small values. Qualitatively, this supports the fact that four is the correct number of change points.

In practice, the Bayesian information criterion (BIC) is commonly used in change point detection to determine the number of change points [166, 170]. It is a model selection procedure that consists in minimizing a constrained likelihood function. In the context of piecewise constant signals with white Gaussian noise, the BIC of the sequential gCPD estimates is

$$\text{BIC}(k) = \left\|\widehat{R}^{(k)}\right\|^2 + k\sigma^2 d \log T \tag{6.2}$$

where k is the step number (as well as the number of change points). The model with lowest BIC is preferred. On the *Gait* data set, the BIC values reported in Table 6.7. The minimum value is 639.95 and is reached for $k = 4$. Moreover, it substantiates the fact that gCPD can accommodate a standard model selection procedure even if it is only an approximation of the optimal signal segmentation.

Part III

Supervised change point detection

7

Calibrating the smoothing parameter through supervised learning

Contents

Abstract

The objective of this chapter is to design an automatic procedure to calibrate detection algorithms when the number of changes is unknown. To that end, we introduce `Alpin` (Adaptive Linear Penalty INference), a supervised approach that learns the smoothing parameter of a linear penalty, using a training set of annotated signals. This procedure consists in minimizing a particular loss function, the excess risk. An efficient implementation is provided, which leads to a linear (in the number of training samples) complexity. Compared to other calibration heuristics, which are restricted to mean-shift detection, `Alpin` can accommodate arbitrary types of changes.

1 Penalized change point detection model

1.1 Problem formulation

This work focuses on the detection of an *unknown* number of change points present in a signal $y = \{y_t\}_{t=1}^{T}$. As described in Section 4, detection amounts to solving the following discrete optimization problem:

$$\widehat{\mathcal{T}}_\beta := \arg\min_{\mathcal{T}} \; R_\beta(\mathcal{T}, y) \tag{7.1}$$

where $R_\beta(\mathcal{T}, y)$ denotes the linearly penalized sum of costs (also referred to as penalized risk) and is given by

$$R_\beta(\mathcal{T}, y) := \sum_{k=0}^{K} c(y_{t_k..t_{k+1}}) \; + \; \beta|\mathcal{T}|. \tag{7.2}$$

with $c(\cdot)$ a user-defined cost function. The estimate $\widehat{\mathcal{T}}_\beta$ is also referred to as the β-optimal estimate. The smoothing parameter $\beta > 0$ controls the trade-off between complexity and goodness-of-fit (measured by the sum of costs). Intuitively, low values of β favour partitions with many regimes and high values of β discard most change points.
In practice, the manual tuning of the smoothing parameter is more art than science, especially when the data are noisy, do not fit a standard model or when the end user has particular expectations (for example detecting only change points of a certain magnitude). A number of off-the-shelf procedures are available in the literature. Arguably, the most well-known are AIC, BIC and Mallows' C_p [122, 144, 166]. They fall into the category of parametric penalties. Such penalties usually stem from model selection considerations. Closed-form expressions are derived, which depend on key parameters of the signal, such as the number of samples and the noise variance [134, 166], the autoregressive order [41], the 2^{nd} order structure [110], etc. Several authors have proposed data driven heuristics to calibrate the smoothing parameter [5, 69, 108]. A general principle is to compute the optimal segmentations for a large of number K of change points and retrospectively select the appropriate one. For instance the heuristics found in Lavielle (2005) uses a threshold on the computed sums of costs. Details about the standard procedures to calibrate the smoothing parameter can be found in Section 4 on page 72.
In this chapter we propose a procedure to calibrate the smoothing parameter using annotated signals.

1.2 Related work

To our knowledge, Hocking et al. (2013) is the only calibration strategy based on supervised learning, in the literature. The authors take advantage of a set of annotated signals to find an appropriate penalty, that can then be applied on new signals. This article is focused on the detection of mean-shifts in DNA data, with the c_{L_2} cost function. The originality of this approach lies in the mapping between the expert annotations and the annotation error, defined as the difference between the estimated number of changes and the true number of changes. Since the annotation error is not convex and cannot be optimized as such, the authors propose to use a convex relaxation, which requires the computation of a large number of segmentations. The approach we propose aims at generalizing this procedure to arbitrary cost functions (and therefore arbitrary types of change).

1.3 Contributions of the chapter

We propose a supervised procedure that learns from a set of annotated training signals an appropriate smoothing parameter for a linear penalty. Our method relies on a convex loss function, that leads to an efficient implementation of the learning step. Arbitrary cost functions can be accommodated under this setting.

2 Properties of the excess risk

This section presents the excess risk, which is later used as a loss function for our supervised procedure. To that end, consider a training set of L annotated signals $\{(y^{(l)}, \mathcal{T}^{(l)}) \,|\, l = 1, \dots, L\}$. Each signal $y^{(l)}$ has $T^{(l)}$ samples and is associated with a segmentation annotation $\mathcal{T}^{(l)} = \{t_1^{(l)}, t_2^{(l)}, \dots\}$. (The annotation $\mathcal{T}^{(l)}$ contains the positions of the change points regarded as "optimal" by an expert.) For a given annotated signal $y^{(l)}$, the excess risk $\mathcal{E}^{(l)}(\beta)$ measures the error between the estimated segmentation $\widehat{\mathcal{T}}_\beta^{(l)}$ (7.7) and the annotation $\mathcal{T}^{(l)}$.

Definition 7.1 (Excess risk). *For a given training signal $y^{(l)}$ and its annotation $\mathcal{T}^{(l)}$, the (empirical) excess risk $\mathcal{E}(\cdot, y^{(l)}, \mathcal{T}^{(l)})$ is the difference between the empirical risks of the expert manual segmentation and the β-optimal estimate, i.e.*

$$\mathcal{E}^{(l)}(\beta) := R_\beta(\mathcal{T}^{(l)}, y^{(l)}) - \underbrace{\min_{\mathcal{T}} R_\beta(\mathcal{T}, y^{(l)})}_{R_\beta(\widehat{\mathcal{T}}_\beta, y^{(l)})}. \tag{7.3}$$

By design, the excess risk is always non-negative. Also, if the smoothing parameter β is such that the excess risk is minimum and equal to 0, then the β-estimate $\widehat{\mathcal{T}}_\beta^{(l)}$ and the manual segmentation $\mathcal{T}^{(l)}$ coincide.

Theorem 7.1 (Convexity). *For any training signal $y^{(l)}$, the function $\beta \mapsto \mathcal{E}^{(l)}(\beta)$ is convex.*

Proof. First note that

$$\min_{\mathcal{T}} R_\beta(\mathcal{T}, y^{(l)}) = \min_{\mathcal{T}} \left[\sum_{k=0}^{K} c(y^{(l)}_{t_k..t_{k+1}}) + \beta|\mathcal{T}| \right] = \min_{K} \left[\min_{|\mathcal{T}|=K} \sum_{k=0}^{K} c(y^{(l)}_{t_k..t_{k+1}}) + \beta K \right] \tag{7.4}$$

It follows that the function $\beta \mapsto \min_{\mathcal{T}} R_\beta(\mathcal{T}, y^{(l)})$ is equal to the pointwise minimum of affine functions, and therefore concave. Since the excess risk (7.3) is an affine function minus a concave function, the function $\beta \mapsto \mathcal{E}^{(l)}(\beta)$ is convex. □

Proposition 7.1 (First derivative of the excess risk). *For a given training signal $y^{(l)}$, the first derivative w.r.t. the parameter β of $\beta \mapsto \mathcal{E}(\cdot, y^{(l)})$ is given by*

$$\frac{\mathrm{d}}{\mathrm{d}\beta} \mathcal{E}(\beta, y^{(l)}) = |\mathcal{T}^{(l)}| - |\widehat{\mathcal{T}}_\beta^{(l)}|. \tag{7.5}$$

Proof. From Equation (7.4), it follows that the function

$$\beta \mapsto \min_{\mathcal{T}} R_\beta(\mathcal{T}, y^{(l)}) \tag{7.6}$$

is the pointwise minimum of affine functions and therefore piecewise affine. Let $[\beta_{min}, \beta_{max}]$ be an interval on which it is affine The first derivative (w.r.t. β) of this function on the interval $[\beta_{min}, \beta_{max}]$ is $|\widehat{\mathcal{T}}_\beta^{(l)}|$, the number of change points in the β-optimal segmentation. In addition, the first derivative of $R_\beta(\mathcal{T}^{(l)}, y^{(l)})$ is clearly $|\mathcal{T}^{(l)}|$. From this, it is straightforward to conclude the proof. □

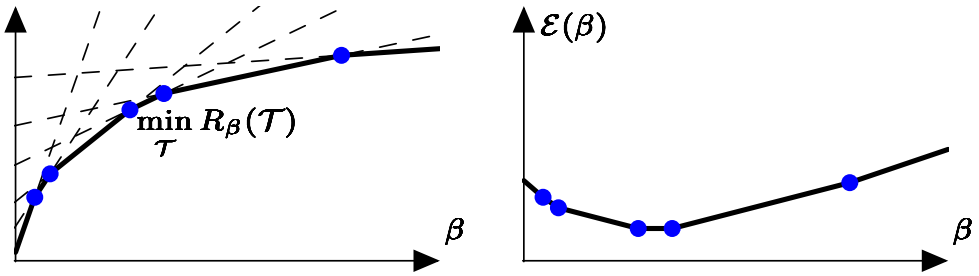


Figure 7.1: (Left) For a given signal $y \in \mathbb{R}^n$ the minimum empirical risk over all change point sets $\mathcal{T}$ is plotted versus β. Following (7.4), the penalized risk is the pointwise minimum of affine functions, marked in dashed lines. (Right) The corresponding excess penalized risk is plotted versus β.

For illustration purposes, a view of the excess risk of a signal is shown on Figure 7.1. In particular, it is shown that the risk of the β-optimal segmentation is the pointwise minimum of affine functions, as demonstrated in Equation (7.4). The convex piecewise affine structure of the excess risk is also visible. The interval on which it is minimum is the interval of optimal smoothing parameter values.

3 Adaptive Linear Penalty INference: the `Alpin` algorithm

In this section, we introduce `Alpin` (Adaptive Linear Penalty INference), which aims at learning an appropriate smoothing parameter $\hat{\beta}$ from the training set. The final objective is to be able to detect an *unknown* number of change points in a new signal, following the same detection strategy as the expert.

Optimization problem. `Alpin` is a supervised method that relies on a loss function to compare the manual segmentations and the estimated segmentations. The optimal smoothing parameter $\hat{\beta}$ is the minimizer of the loss function over the training set, i.e.

$$\hat{\beta} := \arg\min_{\beta > 0} \sum_{l=1}^{L} \mathcal{E}^{(l)}(\beta). \tag{7.7}$$

where the loss function $\mathcal{E}^{(l)}(\beta)$ is the excess risk. By choosing this loss function, our learning procedure aims at finding the smoothing parameter such that the risk of the

corresponding estimator is as close as possible to the risk of the annotation. Thanks to Theorem 7.1 and Proposition 7.1, general-purpose solvers can be used to estimate the optimal smoothing parameter $\hat{\beta}$.

Algorithm 7.1 `Alpin`

Input: annotated signals $y^{(l)}$, associated segmentations $\mathcal{T}^{(l)}$ $(l = 1, \dots, L)$, initial value β_0, step size $\delta_0 \in (0,1)$.
$\beta \leftarrow \beta_0$, $\delta \leftarrow \delta_0$,
repeat
  $g \leftarrow 0$ ▷ Gradient
  for $l = 1, \dots, L$ **do**
    $\widehat{\mathcal{T}} \leftarrow \arg\min_{\mathcal{T}} R_\beta(\mathcal{T}, y^{(l)})$ ▷ Use `Pelt` [98]
    $g \leftarrow g + |\mathcal{T}^{(l)}| - |\widehat{\mathcal{T}}|$
  end for
  $\Delta\beta \leftarrow \delta \times g$
  $\delta \leftarrow \delta \times \delta_0$
  $\beta \leftarrow \beta - \Delta\beta$
until $|\Delta\beta| <$ tolerance
Output: optimal penalty level β.

Computational aspects. Practically, `Alpin` works as follows.

- The algorithm is initialized by randomly picking a signal from the training set. The value of β that minimizes its excess penalized risk is found and then used as a warm start for the optimization of the loss functions (7.7).
- We use a simple gradient descent to minimize the convex loss function $\mathcal{L}(\cdot)$ [34]. To evaluate the loss function $\mathcal{L}(\beta)$ at each step of the optimization method, one has to solve L penalized change point detection problems (7.2). To that end, we use `Pelt` to find the exact β-optimal segmentations $\widehat{\mathcal{T}}_\beta^{(l)}$.

The complexity of `Alpin` is driven by the computation of L β-optimal segmentations, which is performed is linear time, thanks to `Pelt` [98]. Overall, the complexity of one optimization step is $\mathcal{O}(\sum_l T^{(l)})$. Also, since the contributions of each signal $y^{(l)}$ to the loss are independent, all calculations can be done in parallel. Comparatively, "Hocking, 2013" computes for each training signal the optimal segmentations with K change points (K ranging from 1 to K_{max}), resulting in a complexity of the order of $\mathcal{O}(K_{max}(\sum_l T^{(l)})^2)$. `Alpin` is described in Algorithm 7.1.

4 Experiments

4.1 Setting

In this section, `Alpin` is compared to an off-the-shelf statistical penalty"BIC" [166], another supervised method "Hocking, 2013" [82] and a data-driven heuristics "Lavielle, 2005" [108]. All performances are measured with the metrics Hausdorff, RandIndex, F1 score and

AnnotationError. In all experiments, supervised methods (`Alpin` and "Hocking, 2013") are evaluated with cross-validation: each data set is divided into ten folds. Each fold serves once as a training set to learn a smoothing parameter, which is then used to segment the signals of the other nine folds.

4.2 *MeanShift* data set

MeanShift	Metric	`Alpin`	BIC	Hocking, 2013	Lavielle, 2005	`Opt` (c_{L_2})
Scenario 1	Hausdorff	3.99 (±1.11)	3.99 (±1.11)	4.01 (±1.13)	3.99 (±1.11)	0.08 (±0.27)
	RandIndex	0.98 (±0.01)	0.98 (±0.01)	0.98 (±0.01)	0.98 (±0.01)	1.00 (±0.00)
	F1 score	1.00 (±0.01)	1.00 (±0.00)	1.00 (±0.02)	1.00 (±0.00)	1.00 (±0.00)
	AnnotationError	0.00 (±0.05)	0.00 (±0.00)	0.03 (±0.17)	0.00 (±0.00)	–
Scenario 2	Hausdorff	8.34 (±10.61)	96.42 (±24.86)	9.05 (±11.34)	58.06 (±18.43)	4.29 (±3.61)
	RandIndex	0.98 (±0.01)	0.87 (±0.05)	0.98 (±0.01)	0.94 (±0.01)	0.99 (±0.01)
	F1 score	0.96 (±0.09)	0.72 (±0.14)	0.96 (±0.09)	0.65 (±0.06)	0.97 (±0.10)
	AnnotationError	0.08 (±0.31)	1.52 (±0.59)	0.12 (±0.38)	4.01 (±0.10)	–
Scenario 3	Hausdorff	4.30 (±0.95)	4.28 (±0.94)	4.32 (±0.97)	4.28 (±0.94)	0.13 (±0.34)
	RandIndex	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)	1.00 (±0.00)
	F1 score	1.00 (±0.02)	1.00 (±0.00)	1.00 (±0.02)	1.00 (±0.00)	1.00 (±0.00)
	AnnotationError	0.02 (±0.14)	0.00 (±0.00)	0.04 (±0.20)	0.00 (±0.00)	–
Scenario 4	Hausdorff	5.80 (±7.49)	5.49 (±2.44)	10.01 (±24.39)	65.50 (±89.57)	3.14 (±2.60)
	RandIndex	1.00 (±0.00)	1.00 (±0.00)	0.99 (±0.00)	0.99 (±0.01)	1.00 (±0.00)
	F1 score	1.00 (±0.01)	1.00 (±0.00)	0.99 (±0.02)	0.93 (±0.06)	1.00 (±0.00)
	AnnotationError	0.00 (±0.05)	0.00 (±0.00)	0.05 (±0.22)	0.63 (±0.54)	–

Table 7.1: Segmentation results on the *MeanShift* data set. Means and standard deviations are shown. Hausdorff is expressed in number of samples. The margin of F1 score is $M = 10$ samples for Scenario 1 and 2, and $M = 20$ samples for Scenario 3 and 4.

All method in this experiments use the c_{L_2} cost function. Recall that *MeanShift* contains 20-dimensional noisy piecewise constant signals; the number of samples T and the noise standard deviation are respectively equal to $(T = 500, \sigma = 1)$, $(500, 3)$, $(2000, 1)$ and $(2000, 3)$ for Scenario 1, 2, 3 and 4. The hardest scenario is Scenario 2 (least points, most noise) and the easiest is Scenario 3 (most points, lowest noise). Results are summarized in Table 7.1. Several observations can be made from this experiment:

- Supervised methods (`Alpin` and "Hocking, 2013") always outperform the heuristics "Lavielle, 2005". They also perform significantly better than the standard penalization "BIC" on Scenarios 1 and 2, and slightly better on Scenarios 3 and 4. This demonstrates the usefulness of a learning step.

- The standard penalization "BIC", developed in an asymptotic setting, is markedly affected by a low number of samples and high level of noise. In detail, from Scenario 1 to Scenario 2 ($\sigma = 1 \to \sigma = 3$, fixed T), Hausdorff is multiplied by more than 40 and F1 score decreases from 1 to 0.79; from Scenario 4 to Scenario 2 ($T = 2000 \to T = 500$, fixed σ), Hausdorff is multiplied by approximatively 20 and F1 score decreases by a value of 0.19. This is the largest reduction in performance among all methods. Closer examination indicates that "BIC" has a tendency to over-segment signals, a fact also observed in [63, 154].

- The heuristics "Lavielle, 2005" performs significantly better when the number of samples is high, as demonstrated by the close to optimal F1 score and RandIndex on Scenario 3 and Scenario 4 ($T = 2000$). In addition, the number of predicted change

points is also closer to the true one on Scenarios 3 and 4 (AnnotationError is around 0.1) than on Scenarios 1 and 2 (AnnotationError is around 0.5).

- The two supervised methods (`Alpin` and "Hocking, 2013") have comparable performances on this particular data set. The number of change points is correctly estimated, as evidenced by the AnnotationError equal to 0 for all scenarios. According to Hausdorff, `Alpin` is slightly better than "Hocking, 2013"; in all scenarios, they have at most one sample difference, on average.

The main conclusion from this experiment is that supervision significantly improves detection performance. Even though the *MeanShift* signals verify the assumptions under which "BIC" and "Lavielle, 2003" were introduced, those methods still fare markedly worse that `Alpin` and "Hocking, 2013". This can be explained by the fact that "BIC" is an asymptotic criterion and "Lavielle, 2003" relies on a threshold that is not adaptive.

4.3 *FreqShift* data set

In this second experiment, we illustrate the fact that `Alpin` is able to accommodate arbitrary cost functions, contrary to other methods. The cost function c_{rbf} (which is kernel-based and detects changes in the probability distribution) is applied on the time-frequency representation of signals from *FreqShift*. Results from the standard c_{L_2} (applied on the time-frequency representation) are also provided. We compare our supervised method to the optimal detection method `Opt`. Note that `Opt` knows the true number of change points, contrary to `Alpin`. Results are summarized in Table 7.1. Several observations can

FreqShift	Metric	`Alpin` (c_{L_2})	`Alpin` (c_{rbf})	`Opt` (c_{L_2})	`Opt` (c_{rbf})
-5 dB	Hausdorff	81.04 ($\pm$174.78)	48.04 ($\pm$103.46)	34.12 ($\pm$51.28)	27.98 ($\pm$26.66)
	RandIndex	0.95 ($\pm$0.11)	0.97 ($\pm$0.05)	0.98 ($\pm$0.02)	0.98 ($\pm$0.01)
	F1 score	0.76 ($\pm$0.24)	0.80 ($\pm$0.21)	0.81 ($\pm$0.19)	0.82 ($\pm$0.18)
	AnnotationError	0.27 ($\pm$0.71)	0.13 ($\pm$0.44)	–	–
-1 dB	Hausdorff	22.11 ($\pm$24.87)	17.92 ($\pm$19.23)	13.59 ($\pm$6.83)	12.76 ($\pm$4.91)
	RandIndex	0.98 ($\pm$0.01)	0.99 ($\pm$0.01)	0.99 ($\pm$0.00)	0.99 ($\pm$0.00)
	F1 score	0.88 ($\pm$0.15)	0.94 ($\pm$0.12)	0.97 ($\pm$0.09)	0.97 ($\pm$0.08)
	AnnotationError	0.03 ($\pm$0.16)	0.02 ($\pm$0.14)	–	–
0 dB	Hausdorff	17.02 ($\pm$17.97)	17.64 ($\pm$52.09)	11.88 ($\pm$4.73)	11.45 ($\pm$4.13)
	RandIndex	0.99 ($\pm$0.01)	0.99 ($\pm$0.02)	0.99 ($\pm$0.00)	0.99 ($\pm$0.00)
	F1 score	0.95 ($\pm$0.12)	0.98 ($\pm$0.08)	0.99 ($\pm$0.05)	0.99 ($\pm$0.04)
	AnnotationError	0.01 ($\pm$0.11)	0.01 ($\pm$0.16)	–	–
2 dB	Hausdorff	16.29 ($\pm$53.68)	17.65 ($\pm$18.19)	9.68 ($\pm$4.12)	9.64 ($\pm$3.66)
	RandIndex	0.99 ($\pm$0.03)	0.99 ($\pm$0.01)	0.99 ($\pm$0.00)	0.99 ($\pm$0.00)
	F1 score	0.99 ($\pm$0.06)	0.93 ($\pm$0.11)	0.99 ($\pm$0.04)	0.99 ($\pm$0.03)
	AnnotationError	0.02 ($\pm$0.21)	0.01 ($\pm$0.11)	–	–

Table 7.2: Segmentation performance. Means and standard deviations on the *FreqShift* data set are shown. Hausdorff is expressed in number of samples. The margin of F1 score is $M = 20$ samples. Note that the number of change points is known for `Opt` but not for `Alpin`.

be made from this experiment:

- There are two behaviours, depending on the noise level. For SNRs at -1 dB and -5 dB, `Alpin` (c_{rbf}) is more accurate than `Alpin` (c_{L_2}), according to all metrics. In particular, AnnotationError of `Alpin` (c_{rbf}) is twice as low as AnnotationError

of `Alpin` (c_{L_2}). For SNRs at 0 dB and 2 dB, both methods are close to each other. More precisely, `Alpin` (c_{L_2}) has slightly better HAUSDORFF, and is also slightly worse according to F1 SCORE at 0 dB and ANNOTATIONERROR at 2 dB. This indicates that using the kernel-based cost function makes `Alpin` more robust to noise. When the SNR is above a certain threshold, both cost functions are equivalent. This can be explained by the fact that the time-frequency representations of the signals do not exactly follow the piecewise constant model with Gaussian white noise assumed by c_{L_2}.

- Optimal methods that know the number of change points beforehand always perform better than `Alpin`. The difference however decreases as the SNR increases. As a comparison, the best approximate methods on *FreqShift*, `gkCPD` (see Section 3 on page 116), has approximatively the same metric values as `Alpin` (c_{rbf}). This indicates that, on this particular data set, not knowing the number of change points (but using the exact penalized detection `Pelt`) has the same effect as knowing the number of changes and using the approximate method `gkCPD`.

To sum up, using the kernel-based cost function c_{rbf} makes the supervised approach `Alpin` more robust to noise. This is a significant advantage of `Alpin`: it can accommodate arbitrary cost functions.

4.4 Execution time comparison

	`Alpin`	Hocking, 2013
300 samples	2 minutes	12 minutes
400	3	21
500	7	33
600	12	49
700	9	67
800	23	88
900	21	113
1000	19	141
2000	30	583

Table 7.3: Execution time in minutes vs the number of samples of each of the 100 signals of the data set. All times refer to running a Python implementation of `Alpin` and "Hocking, 2013" on a Linux computer with 24 Intel processors running at 2.80 GHz (CPU).

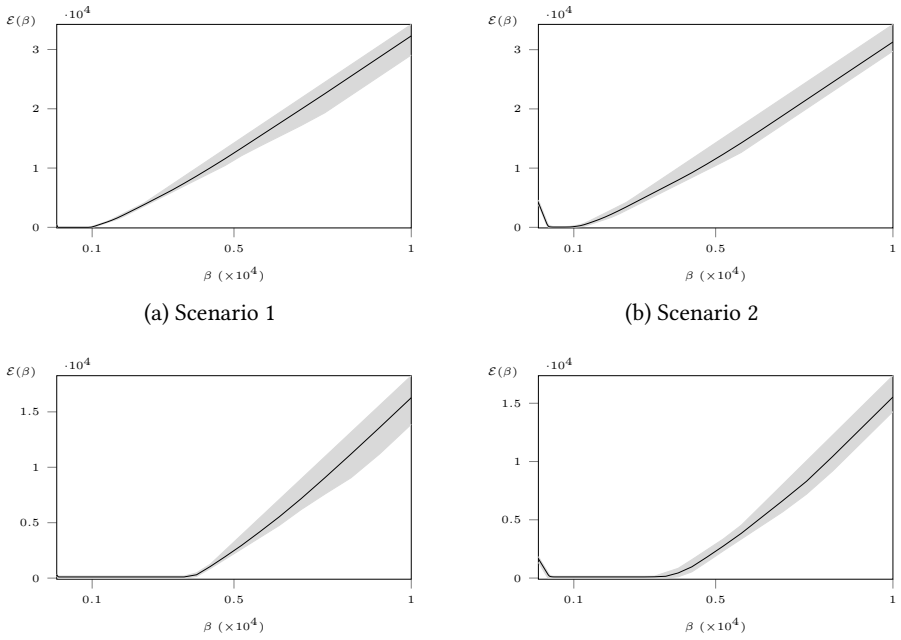
We compare the computation time needed to learn the correct penalty level on noisy univariate signals with a random number of change points (uniformly chosen between 3 and 7) and randomly located. For different numbers $T \in \{300, 400, \dots, 2000\}$ of samples, 100 signal realizations are generated. The algorithms `Alpin` and "Hocking, 2013" are applied to those (annotated) signals with the cost function c_{L_2}. The computation times are reported in Table 7.3. Overall, `Alpin` performs faster than its counterpart. For instance, the execution time for processing 100 signals of length $n = 500$ is 33 minutes for "Hocking, 2013" but only 7 minutes for `Alpin`. The convex excess risk (7.3) and the possibility to directly minimize it with standard optimization methods allows to keep a reasonable computing time, which makes it suitable for real-life situations. From a computational

standpoint, `Alpin` and "Hocking, 2013" have indeed different complexities: the former is linear in the number of samples while the latter is quadratic [82].

5 Discussion

We have shown that `Alpin` can be used to learn the correct penalty level for change point detection and now provide further insights on the algorithm.

5.1 Comments on the excess risk



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Figure 7.2: Average excess risk (7.8) on the *MeanShift* data set, for the c_{L_2} cost function. Minimum and maximum (over the data set's signals) excess risks are given by the gray area.

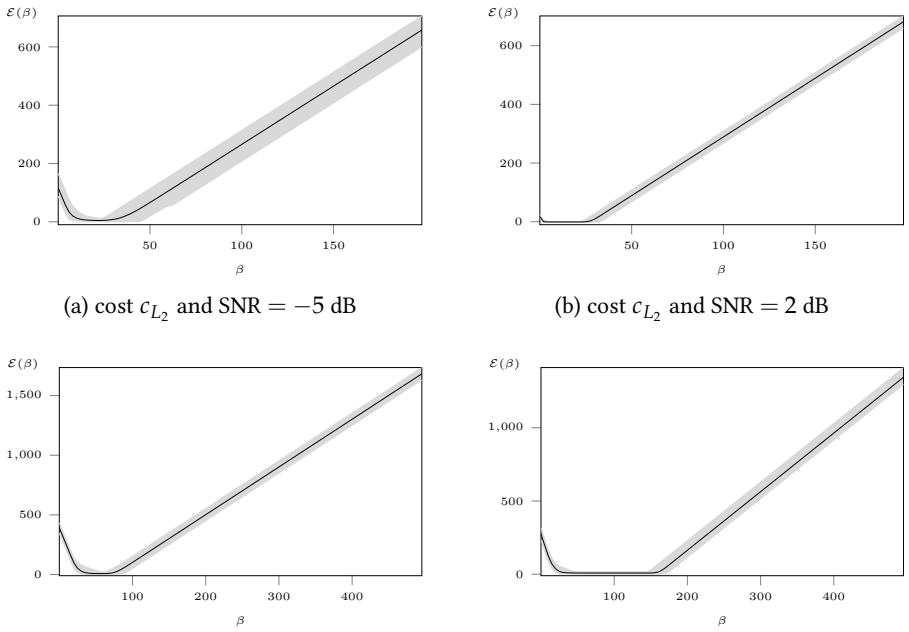
We illustrate the relationship that exists between the shape of the excess risk curve and the difficulty of the segmentation task, on different data sets and for different cost functions. To that end, the curve

$$\beta \mapsto \frac{1}{L} \sum_{l=1}^{L} \mathcal{E}^{(l)}(\beta) \tag{7.8}$$

is plotted for the four scenarios of the *MeanShift* data set (cost function c_{L_2}) and the time-frequency representation of the *FreqShift* data set (cost functions c_{L_2} and c_{rbf}). The excess risk curve on the *MeanShift* data set is displayed on Figure 7.2. Several observations can be made:

- For all scenarios, there is an interval $[\beta_{\min}, \beta_{\max}]$ on which the curve reaches its minimum and is flat. Any value within interval is an optimal penalty value for `Alpin`. A large interval implies that it is easier to find a suitable smoothing parameter value.

- The size of this interval depends on the simulation parameters (T, σ). When the number of samples grows from $T = 500$ (Scenario 1 and Scenario 2) to $T = 2000$ (Scenario 3 and Scenario 4), the right endpoint $\beta_{\max}$ increases from approximatively 0.1×10^4 to approximatively 0.3×10^4, enlarging the optimal interval. When the noise level grows from $\sigma = 1$ (Scenario 1 and Scenario 3) to $\sigma = 3$ (Scenario 2 and Scenario 4), the left endpoint $\beta_{\min}$ increases from approximatively 0.00×10^4 to approximatively 0.03×10^4, shortening the optimal interval. This observation confirms the (well-known) fact that detection is harder when there are less samples and/or more noise.



(a) cost c_{L_2} and SNR $= -5$ dB

(b) cost c_{L_2} and SNR $= 2$ dB

(c) cost c_{rbf} and SNR $= -5$ dB

(d) cost c_{rbf} and SNR $= 2$ dB

Figure 7.3: Average excess risk (7.8) on the *FreqShift* data set. Minimum and maximum (over the data set's signals) excess risks are given by the gray area.

The excess risk curve on the *FreqShift* data set is displayed on Figure 7.3. Several observations can be made:

- Again, an interval on which the excess risk is flat exists. The length of this interval depends on the cost function and the noise level. For both cost functions, it increases when the SNR increases, as expected: the easier the task, the easier it is to find a suitable smoothing parameter value. Also, more variations are observed when SNR $= -5$ dB. The c_{rbf} cost function yields a longer interval for both noise levels. This indicates that using a kernel make the penalty learning problem easier to solve, compared to c_{L_2}.

- Depending on the cost function, the optimal smoothing parameter value can take different values, illustrating the usefulness of a procedure to learn it.

5.2 Double labels

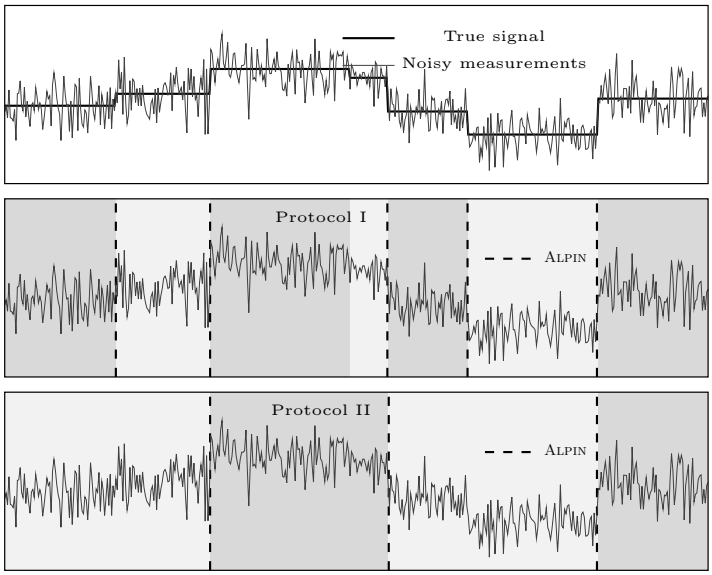


Figure 7.4: (Top) Signal example and its noisy version ($T = 500, \sigma = 2$). (Middle) Prediction and expert partition according to Protocol I. (Bottom) Prediction and expert partition according to Protocol II.

Different annotations on the same signals can arise when experts are not interested in the same physiological phenomena, which result in distinct segmentation strategies. In this situation, the optimal smoothing parameter should depend on the segmentation strategy of the experts. To illustrate how `Alpin` adapts to two different annotations on the same signals, a synthetic data set is constructed as follows. A set of 100 piecewise constant functions from $[0,1]$ to $\mathbb{R}$ are simulated, with a number of change points randomly chosen between 3 and 7. The length of each regime is drawn uniformly between 0.05 and 0.3 and the jumps between regimes have random amplitudes between 1 and 5. Those functions are then sampled on an equispaced grid of $T = 500$ points and corrupted by a Gaussian noise of variance $\sigma = 2$. The following two annotation protocols are considered.

- In Protocol I, all change points from the true underlying function are considered.
- In Protocol II, only the biggest and most visible change points are regarded as real change points. More precisely the changes with amplitude below 3 are discarded.

Figure 7.4 presents an example of the results obtained with `Alpin` by using only train data from Protocol I or only train data from Protocol II. Depending on the protocol, the number of detected change points is different, although the input signal is exactly the same. This illustrates that the algorithm is able to learn that Protocol II only considers the largest changes. In Protocol I, one change point is missed (middle plot) which is due to the fact

that the mean-shift is small, a situation that is rarely found in the training database. This experiment illustrates the ability of `Alpin` to adapt to different segmentation strategies.

6 Conclusion

In this chapter, an automatic procedure to calibrate detection algorithms when the number of changes is unknown, has been described. This supervised method, denoted `Alpin`, learns from a set of annotated training signals an appropriate smoothing parameter for a linear penalty. In a nutshell, the optimal smoothing parameter is the minimizer of a suitable loss function. This loss function can accommodate arbitrary cost functions, and is shown to be convex, which leads to an efficient implementation of the learning step, with linear complexity. Numerical experiments on synthetic and real-world data sets show that `Alpin` outperforms standard non-supervised penalization methods. In addition, compared to other supervised approaches, `Alpin` is faster and can be applied to detect arbitrary types of changes. Another benefit of `Alpin` is its ability to adapt to different annotation strategies, even when the signals are the same. This is illustrated in the last section of this chapter, and would be interesting to further investigate in future research.

8

Metric learning for change point detection

Contents

Abstract

The objective of this chapter is to design an automatic procedure to calibrate the cost function of change point detection algorithms. To that end, we introduce a scheme to convert signal annotations into similarity/dissimilarity constraints. A non-parametric transformation of the signal samples is then learned to enforce those constraints. This procedure relies on a kernel metric learning algorithm, and can accommodate full or partial annotations. Compared to all methods tested in this manuscript, this supervised approach achieves the best performance on the *Gait* data set, without any pre-processing.

1 Change point detection with a Mahalanobis-type pseudo-norm

1.1 Problem formulation

We are interested in detecting multiple change points in an $\mathbb{R}^d$-valued signal $y = \{y_t\}_{t=1}^T$, using a general class of non-parametric cost functions $c_{\mathcal{H},M}$ based on a kernel Mahalanobis-type pseudo-norm $\|\cdot\|_{\mathcal{H},M}$. The cost function $c_{\mathcal{H},M}$ is formally defined as follows. Let $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ denote a kernel function and $\mathcal{H}$, the associated reproducing Hilbert space (RKHS). The related mapping function $\phi : \mathbb{R}^d \to \mathcal{H}$ is implicitly defined by $\phi(y_t) = k(y_t,\cdot) \in \mathcal{H}$ and $\langle \phi(y_s)|\phi(y_t)\rangle_{\mathcal{H}} = k(y_s,y_t)$. The RKHS norm $\|\cdot\|_{\mathcal{H}}$ is also implicitly defined by $\|\phi(y_t)\|_{\mathcal{H}}^2 = k(y_t,y_t)$. The cost function $c_{\mathcal{H},M}$ is given by

$$c_{\mathcal{H},M}(y_{a..b}) := \sum_{t=a+1}^{b} \|\phi(y_t) - \bar{\mu}_{a..b}\|_{\mathcal{H},M}^2 \quad (0 \leq a < b \leq T) \tag{8.1}$$

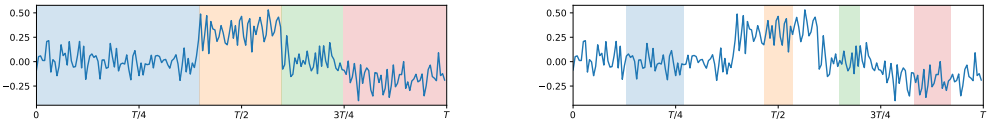
where $\bar{\mu}_{a..b}$ is the mean value of the embedded sub-signal $\{\phi(y_t)\}_{t=a+1}^b$ and $\|\cdot\|_{\mathcal{H},M}$ denotes the Mahalanobis-type (pseudo-)norm. This norm $\|\cdot\|_{\mathcal{H},M}$ is entirely determined by the positive definite matrix M through the following relation:

$$\|\phi(y_t) - \phi(y_s)\|_{\mathcal{H},M}^2 := (\phi(y_t) - \phi(y_s))' M (\phi(y_t) - \phi(y_s)). \tag{8.2}$$

In this context, change point detection amounts to minimizing the following sum of costs:

$$V(\mathcal{T}) := \sum_{k=0}^{|\mathcal{T}|} c_{\mathcal{H},M}(y_{t_k..t_{k+1}}) \tag{8.3}$$

under certain constraints on the complexity of the segmentation $\mathcal{T}$, depending on whether the number of changes is known or not (more details in Section 4 on page 72). In general, the cost function controls the type of change point that can be detected. In practice, engineering an appropriate cost function is a manual process (trial and error) that requires prior knowledge on the underlying phenomena present in the signals. We refer the reader to Section 2 on page 53 where a number of off-the-shelf cost functions are described. However, the end-user is not always able to translate his prior knowledge into an appropriate cost function, or the standard costs might not be able to correctly model the data. In this context, a procedure to automatically design the cost function is needed. In this chapter, we propose to calibrate the cost function using annotated signals. Two type of annotations are considered: fully annotated signals and partially annotated signals. A signal is said to be *fully annotated* if the indexes of target change points are provided. A signal is said to be *partially annotated* if the start and end indexes of a portion of each regime are provided (but the change point locations are not). The two types are illustrated in Figure 8.1. The objective in both situations is to be able to reproduce the segmentation strategy of the annotations. (Note that in Chapter 7, the smoothing parameter is learned using fully annotated signals.)



(a) Full labels.

(b) Partial labels.

Figure 8.1: Annotation types: (a) the full segmentation is available and (b) only the partial segmentation is available. The coloured areas are homogeneous sub-signals provided by an expert. The signal is the same in both situations.

1.2 Related work

To the best our knowledge, there is only one work on change point detection that is based on a supervised strategy and aims at finding an appropriate cost function [105]. The authors consider a *linear* Mahalanobis metric. They propose a learning strategy to accommodate partial labels: partial annotations are completed using change point detection with an initial cost function, then a metric is learned, the partial annotations are completed again with the new cost function, and so on. According to the authors, this non-convex procedure relies on the ability of the user to properly initialize the cost function. Also, for full or partial annotations, `Opt` is applied repeatedly during the learning step.
Our problem falls into the category of metric learning, which was first introduced in the context of classification. The goal for those methods is to learn from a training set a task-specific similarity measure to have better testing performance than off-the-shelf measures such as Euclidean distance or cosine similarity. Most works focus on Mahalanobis-type distances and adapt the metric matrix using information from the training samples: similarity/dissimilarity constraints for pairs of samples [55, 149, 165], relative constraints for triplet of samples [85, 143]. In our context, a major drawback of those methods is that a *linear* Mahalanobis cost function is only sensitive to mean-shifts. A related topic is kernel learning. Instead of learning a metric matrix, those methods aims at learning a kernel matrix over the data. This has the advantage of defining a non-linear mapping but similarity between out-of-sample points (i.e. not present in the training set) cannot be computed. This setting, called transductive, where all samples (labelled and unlabelled) are provided at the beginning of the learning step, is not adapted to our context.
In this chapter, we transpose the kernel-based approach of [87] to the context of change point detection, which offers a new perspective on supervised signal segmentation. This method relies on the kernelization results on the LogDet regularization [103]. In addition the resulting algorithm is easy to implement and contrary to kernel learning methods, the learned metric can be applied to unseen data.

1.3 Contributions of the chapter

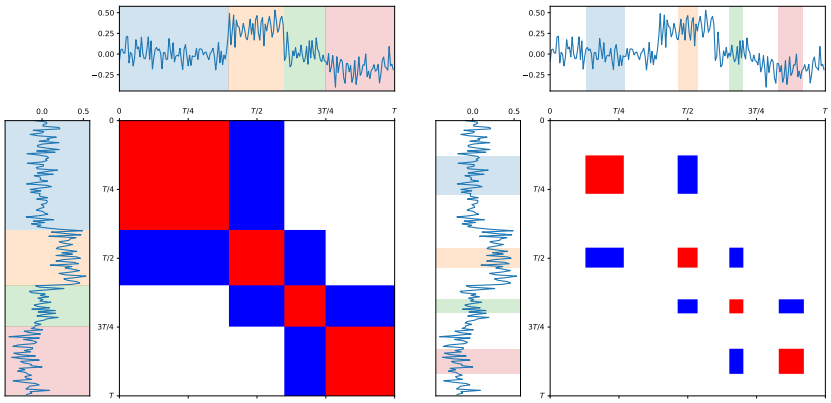
The contribution of this chapter is a method to adapt a cost function to a set of training examples (i.e. signals and their manual segmentations). Two types of annotations can be used to learn the appropriate cost function: full or partial. Compared other supervised methods, our approach is more general because any kernel can be used, not just the linear kernel. Once learned, this cost function can be combined with a search method (`Opt`, `Win`, gkCPD, etc.) and a complexity constraint (fixed number of changes, linear penalty,

`Alpin`, etc.) to create a change point detection algorithm that is able to reproduce the segmentation strategy of the annotations. The learned algorithm can be applied on any new and unseen signal (i.e. not in the training set) sharing the same properties or on a partially annotated signal.

2 Metric learning with a kernel-based approach

Consider a set of L annotated signals $y^{(l)}$ ($L = 1\dots, L$). The signal $y^{(l)}$ can either be fully annotated or partially annotated. Our approach consists in (i) a learning step, during which an optimal metric matrix $\widehat{M}$ is estimated and (ii) a predicting step, during which change point detection is performed on signals using the learned cost function $c_{\mathcal{H},\widehat{M}}$. To detect change points, any search method (for instance, `gkCPD`, `Opt`, etc.) and any complexity constraint (fixed number of changes, linear penalty, `Alpin`, etc.) can be used in combination with the learned cost function $c_{\mathcal{H},\widehat{M}}$.

2.1 From labels to constraints



(a) Full labels. (b) Partial labels.

Figure 8.2: Illustration of the scheme to transform labels into constraints. Similarity/dissimilarity constraints can be stored in a matrix A. For a signal y, the coefficient A_{st} is equal to $+1$ is y_s and y_t are similar, -1 is y_s and y_t are dissimilar, and 0 otherwise. The matrix A is displayed when (a) the full segmentation is available and (b) only the partial segmentation is available. Annotations are highlighted in coloured areas. Red is for $+1$ (similar), blue for -1 (dissimilar) and white for 0 (no relation). The signal is the same in both situations.

In the metric learning literature, algorithms rely on constraints to learn the desired norm [85, 165]. More precisely, *similarity* constraints are pairs of samples that should be close according to the learned distance. Conversely, *dissimilarity* constraints are pairs of samples that should be far according to the learned distance. In the context of supervised classification, those constraints are derived from the class labels of the samples. Such information are not available in our setting. Fortunately, we are able to construct similarity/dissimilarity constraints from the full or partial signal annotations, using the following

scheme. Two samples $y_s^{(l)}$ and $y_t^{(l)}$ from a training signal $y^{(l)}$ are considered "similar" if they belong to the same regime and "dissimilar" if they belong to two consecutive regimes. Pairs of samples that are neither from the same regime nor two consecutive regimes do not create any similarity/dissimilarity constraint. Also, in the situation of partial labels, samples that do not belong to a homogeneous portion of the signal (according to the annotation) do not create any constraint. Thanks to this scheme, one does not need to know the nature of the regimes to construct constraints. Both situations (full and partial labels) are illustrated in Figure 8.2.

2.2 Kernel metric learning

Our method relies on a kernel metric learning procedure, proposed by [87] and briefly presented in this section for completeness. This particular method was introduced in the context of supervised classification, with a kernel to accommodate problems with non-linear decision boundaries. The learned metric matrix $\widehat{M}$ is the solution to the following constrained optimization problem:

$$\begin{aligned} \min_{M \succeq 0} \quad & D_{LD}(M, I) \\ \text{s.t.} \quad & \left\| \phi(y_s^{(l)}) - \phi(y_t) \right\|_{\mathcal{H},M}^2 \leq u, \; y_s^{(l)} \text{ and } y_t \text{ similar samples} \\ & \left\| \phi(y_s^{(l)}) - \phi(y_t) \right\|_{\mathcal{H},M}^2 \geq v, \; y_s^{(l)} \text{ and } y_t \text{ dissimilar samples} \end{aligned} \tag{8.4}$$

where

$$D_{LD}(M, M_0) := \operatorname{tr}(MM_0^{-1}) - \log\det(MM_0^{-1}) \tag{8.5}$$

is the LogDet divergence which acts as a distance on the set of symmetric positive definite matrices and $u > 0$ (resp. $v > 0$) is an upper (resp. lower) bound on the intra-regime (resp. inter-regime) pairwise distances. The distance between M and the identity matrix is akin to a regularization. However, in this formulation, optimization is performed in the space of positive semi-definite matrices on the feature space $\mathcal{H}$, which is possibly infinite dimensional and only implicitly defined through the kernel $k(\cdot, \cdot)$. The authors have proven that this problem has an equivalent finite-dimensional formulation, where the optimization is performed on the Gram matrices of the data. Under this setting, the output of the kernel metric learning algorithm is *not the optimal metric matrix* $\widehat{M}$ but rather the matrix

$$\widehat{G} := \phi(y_s^{(l)})' \widehat{M} \phi(y_t^{(l)}) \tag{8.6}$$

containing the inner-products for all training samples $y_s^{(l)}$ and $y_t^{(l)}$, i.e. samples that belong to an annotated portion of a training signal. The upper and lower bounds for the similarity and dissimilarity constraints are empirically set to as the 1 and 99-th percentiles of the distribution of pairwise distances.

2.3 Computing the learned cost function

After the metric learning step, all that is left is to combine the associated cost function $c_{\widehat{M},\mathcal{H}}(\cdot)$ with a search method (`Opt`, `gkCPD`, `Win`,...) and a complexity penalty (fixed K, linear penalty, ...) in order to perform change point detection. To that end, one must be

able to compute the cost function on any given signal. After simple manipulations, for a given signal $z = \{z_t\}_{t=1}^T$, the learned cost function on the sub-signal $z_{a..b}$ $(0 \leq a < b \leq T)$ is equal to

$$c_{\mathcal{H},\widehat{M}}(z_{a..b}) = \sum_{t=a+1}^{b} \phi(z_t)'\widehat{M}\phi(z_t) - \frac{1}{(b-a)} \sum_{s,t=a+1}^{b} \phi(z_s)'\widehat{M}\phi(z_t). \tag{8.7}$$

The cost function only depends on the matrix $\widehat{M}$ through the inner-products $\phi(z_t)'\widehat{M}\phi(z_t)$. A difficulty lies in the fact that the metric matrix $\widehat{M}$ is not explicitly available. Two scenarios exist: (i) both samples belong to the training set or (ii) at least one sample is not in the training set. When the signal z to segment is one of the training signals $y^{(l)}$ and has only been partially annotated (in which case, we are performing segmentation completion), the two scenarios can occur. When the signal z to segment is new and unseen, we are exclusively in the scenario (ii). Scenario (i) is easily solved because the output of the metric learning step is the values of all inner-products of pairs of training samples. Thanks to a representer type of theorem [87, Theorem 1], this also proves to be enough to compute the inner-products in scenario (ii). More precisely, the following expression can be used for any samples z_s and z_t:

$$\phi(z_s)'\widehat{M}\phi(z_t) = k(z_s, z_t) + k_s' G^{-1}(\widehat{G} - G)G^{-1}k_t \tag{8.8}$$

where

- $k_\bullet$ denotes the column vector $[k(z_\bullet, y_1), k(z_\bullet, y_2), \dots]'$ with $\{y_1, y_2, \dots\}$ the set of training samples;
- G is the matrix of inner-products of the training samples in the untransformed space, i.e. $G_{st} := k(y_s, y_t)$;
- $\widehat{G}$ is defined in (8.6).

2.4 Intuition behind the cost function $c_{\mathcal{H},M}$

Using the cost function $c_{\mathcal{H},M}$ can be seen as performing the following operations: the signal samples are first mapped to a high-dimensional feature space (through ϕ) then they are linearly transformed, then mean-shifts are detected. Indeed, decomposing the symmetric matrix $M = U'U$ yields

$$\|\phi(y_t) - \phi(y_s)\|_{\mathcal{H},M}^2 = \|U\phi(y_t) - U\phi(y_s)\|_{\mathcal{H}}^2. \tag{8.9}$$

Therefore, measuring distances (in the feature space) with the pseudo-norm $\|\cdot\|_{\mathcal{H},M}$ is equivalent to applying a transformation $U\phi(\cdot)$ on the data. The resulting sum-of-cost function $V(\cdot)$ measures the error when approximating the transformed signal $\{U\phi(y_t)\}_t$ by a piecewise constant function. The first mapping ϕ is unsupervised (i.e. not task-specific) and extracts a great number (possibly infinite) of features while the second mapping U is linear and task-specific. This setting has two major advantages. If the chosen kernel implicitly defines an infinite-dimensional RKHS, the transformation, determined by M (also infinite dimensional) is non-parametric. Also, it is non-linear which is necessary in most real-world contexts, and essential for our physiological signals (for instance the *Gait*

data set whose changes are not simple mean-shifts).
Note that $c_{\mathcal{H},M}$ generalizes some well-known cost functions. If $k(\cdot,\cdot)$ is set to the linear kernel and $M = Id$, then $c_{\mathcal{H},M}$ is formally equivalent to the cost function c_{L_2}. For any kernel function, and $M = Id$, $c_{\mathcal{H},M}$ is formally equivalent to c_{kernel} (see Section 2.2.3). In particular, if $k(\cdot,\cdot)$ is set to the Gaussian kernel (and $M = Id$), then $c_{\mathcal{H},M}$ is formally equivalent to the cost function c_{rbf} (see Section 2.2.3).

3 Experiments

The performance of our supervised strategy is compared to standard (unsupervised) change point detection methods. First, we use the metric learning procedure to learn a cost function from a set of training signals and use it to segment new signals. Next, we use the metric learning procedure to complete partial segmentation annotations.

3.0.1 Experimental setting

We use the kernel metric learning procedure to improve the performance of three change point detection methods: `gkCPD` (greedy segmentation with the c_{rbf} cost function), `Win` (c_{rbf}) (window-based segmentation with the c_{rbf} cost function), and `Win` (c_{L_2}) (window-based segmentation with the c_{L_2} cost function). The supervised algorithms are denoted ♡`gkCPD`, ♡`Win` (c_{rbf}), and ♡`Win` (c_{L_2}). For methods that use the c_{rbf} cost function, a Gaussian kernel is used; for the method that uses the c_{L_2} cost function, a linear kernel is used.
The parameters of the algorithms are calibrated as follows. For all methods, the number of changes is known beforehand. For the Gaussian kernel, the scale parameter is tuned according to the "median heuristics" [142]. Window-based methods use a 100-sample long window. For the metric learning, the similarity/dissimilarity constraints are created from the annotations using 50 samples from each regime. The so-called slack parameter [87] is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ using cross-validation on a random subset of 10 signals. In all experiments, supervised methods (those with a ♡) are evaluated with cross-validation: each data set is divided into ten folds. Each fold serves once as a training set to learn a cost function, which is then used to segment the signals of the other nine folds.

3.1 Supervised segmentation of new signals

In this experiment, an optimal cost function is learned from a training set of signals and used to segment new signals from a testing set.

3.1.1 *FreqShift* data set

SNR	Metric	♡gkCPD	gkCPD	♡Win (c_{rbf})	Win (c_{rbf})	♡Win (c_{L_2})	Win (c_{L_2})
−5 dB	Hausdorff	35.26 (±26.65)	48.28 (±57.06)	260.93 (±150.49)	374.48 (±132.76)	288.08 (±157.90)	368.34 (±137.01)
	RandIndex	0.98 (±0.01)	0.97 (±0.02)	0.91 (±0.06)	0.84 (±0.06)	0.89 (±0.06)	0.84 (±0.07)
	F1 score	0.76 (±0.18)	0.71 (±0.21)	0.59 (±0.22)	0.32 (±0.22)	0.52 (±0.23)	0.35 (±0.23)
−1 dB	Hausdorff	15.60 (±6.16)	17.80 (±7.31)	52.42 (±106.32)	189.09 (±166.78)	53.93 (±106.15)	148.59 (±163.52)
	RandIndex	0.99 (±0.00)	0.98 (±0.01)	0.98 (±0.03)	0.94 (±0.05)	0.97 (±0.04)	0.95 (±0.04)
	F1 score	0.93 (±0.11)	0.90 (±0.13)	0.94 (±0.11)	0.71 (±0.19)	0.88 (±0.16)	0.77 (±0.19)
0 dB	Hausdorff	16.14 (±6.61)	18.67 (±8.38)	45.58 (±98.88)	110.19 (±152.74)	37.71 (±83.73)	97.12 (±148.83)
	RandIndex	0.99 (±0.01)	0.98 (±0.01)	0.98 (±0.02)	0.96 (±0.04)	0.98 (±0.02)	0.97 (±0.04)
	F1 score	0.93 (±0.14)	0.90 (±0.15)	0.94 (±0.11)	0.83 (±0.18)	0.90 (±0.14)	0.86 (±0.16)
2 dB	Hausdorff	14.57 (±4.68)	16.10 (±7.56)	11.60 (±4.08)	20.68 (±39.89)	18.69 (±17.75)	17.84 (±39.04)
	RandIndex	0.99 (±0.00)	0.99 (±0.01)	0.99 (±0.00)	0.99 (±0.01)	0.99 (±0.01)	0.99 (±0.01)
	F1 score	0.95 (±0.11)	0.93 (±0.13)	0.99 (±0.04)	0.92 (±0.14)	0.92 (±0.12)	0.95 (±0.12)

Table 8.1: Performance (means and standard deviations) on the *FreqShift* data set. Margin for F1 score is 20 samples. Hausdorff is in number of samples.

We start off by applying our supervised strategy on signals from the *FreqShift* data set. In this experiment, we use the time-frequency representation of the signals. Results are provided in Table 8.1, from which several observations can be made.

- **Supervision improves segmentation accuracy.** Except in one situation, the supervised method is more accurate than its unsupervised counterpart, according to all metrics. This is particularly visible when SNR $= -1$ dB for `Win` methods: for instance, HAUSDORFF is around t three times lower thanks to supervision. Generally, the amelioration is large for high levels of noise. The difference becomes smaller as the SNR goes to 2 dB, where supervised and unsupervised methods converge, and are equally accurate, to the point that $\texttt{Win}(c_{L_2})$ have better scores than $\heartsuit\texttt{Win}(c_{L_2})$ (one sample difference in HAUSDORFF for instance).

- **Window-based methods are less robust to noise.** Even with supervision, we observe that window-based methods remain less robust to noise. For instance, even though $\heartsuit\texttt{Win}(c_{\text{rbf}})$ and $\heartsuit$gkCPD share the same learned cost function, the greedy segmentation method has better scores for all SNRs, except 2 dB. This was already observed on this data set, see Section 3 on page 116. Interestingly, window-based methods are quite accurate when SNR is high: for instant $\heartsuit\texttt{Win}(c_{\text{rbf}})$ is the best when SNR $= 2$ dB.

To sum up, our kernel metric learning strategy improves segmentation performance on *FreqShift*. It can accommodate either c_{L_2} or c_{rbf} and any search methods that can be combined with a cost function. This illustrates the benefits of learning a suitable metric to detect change points.

3.1.2 *Gait* data set

In this experiment, an optimal cost function is learned from a set of *Gait* signals and used to segment new *Gait* signals. Contrary to previous experiments (Chapter 6), segmentation is performed on *the time domain representation* of the signals and not the the time-frequency representation. The only preprocessing consists in centering and scaling all dimensions of the signal to unit variance. The c_{L_2} cost function is not used here because it can only detect mean-shifts.

Gait data set	$\heartsuit$gkCPD	gkCPD	$\heartsuit\texttt{Win}$ (c_{rbf})	`Win` (c_{rbf})
HAUSDORFF	0.94 ($\pm$0.58)	1.35 ($\pm$1.35)	5.93 ($\pm$2.65)	6.01 ($\pm$2.24)
RANDINDEX	0.93 ($\pm$0.03)	0.92 ($\pm$0.05)	0.84 ($\pm$0.05)	0.83 ($\pm$0.04)
F1 SCORE	0.91 ($\pm$0.15)	0.86 ($\pm$0.18)	0.62 ($\pm$0.16)	0.60 ($\pm$0.15)

Table 8.2: Performance (means and standard deviations) on the *Gait* data set. Margin for F1 SCORE is one second. HAUSDORFF is in seconds.

Results are provided in Table 8.2. Several observations can be made.

- Observations from the previous experiment, on *FreqShift*, are confirmed: supervised methods outperform their unsupervised counterparts, according to all metrics and, even though they share the same learned cost function, $\heartsuit$gkCPD outperforms $\heartsuit\texttt{Win}$ (c_{rbf}).

- The method ♡gkCPD is the most accurate segmentation method presented in this work: in all experiments on the *Gait* data set, it is the only one with its F1 SCORE above 0.90 and its HAUSDORFF under one second (see also Section 4 on page 119 for scores of other methods). Note that ♡gkCPD is applied on the raw signals: the learning step has learned the correct representation from available change point annotations. This removes the need to manually design a representation in which change points are easy to detect.

Those results are promising for the segmentation of *Gait* signals More insights can be gained by looking at the segmentation accuracy by change point type, reported in Table 8.3. This confirms the fact that ♡gkCPD is the best performing method on the *Gait* data set. For instance, error is under one second for all change types. Supervision improves the detection precision for all change points and the error variance is lower, indicating that detection is also more stable.

	♡gkCPD	gkCPD	♡Win (c_{rbf})	Win (c_{rbf})
Stand/Walk	0.47 (±0.42)	0.76 (±0.99)	5.31 (±2.86)	5.28 (±2.53)
Walk/Turnaround	0.61 (±1.23)	1.07 (±2.04)	1.01 (±1.42)	1.12 (±1.33)
Turnaround/Walk	0.94 (±1.83)	1.14 (±2.04)	1.83 (±2.52)	2.40 (±2.59)
Walk/Stand	0.53 (±0.36)	0.62 (±0.42)	1.84 (±2.60)	2.16 (±2.62)

Table 8.3: Accuracy (means and standard deviations) in seconds.

3.2 Segmentation completion

In this experiment, a different task is tackled. A cost function is learned from a partially annotated signal and applied on the same signal to recover the change points. Such a scenario corresponds to a situation when an expert provides only a small part of the underlying segmentation and expect the algorithm to complete the annotation. The algorithm ♡gkCPD is applied on the *Gait* data set. For every signal, portions of $w \in \{0.1, 0.5, 1, 1.5\}$ seconds from each regime are used as partial annotations. Results are reported in Table 8.4.

♡gkCPD	0.1 sec	0.5 sec	1 sec	1.5 sec
HAUSDORFF	1.20 (±1.19)	0.91 (±0.53)	1.37 (±1.32)	1.50 (±1.33)
RANDINDEX	0.92 (±0.04)	0.93 (±0.03)	0.93 (±0.05)	0.93 (±0.04)
F1 SCORE	0.87 (±0.17)	0.90 (±0.16)	0.89 (±0.15)	0.90 (±0.14)

Table 8.4: Accuracy (means and standard deviations) in seconds against the length of well-labelled data. Margin for F1 SCORE is one second. HAUSDORFF is in seconds.

On the *Gait* data set, ♡gkCPD performs best when 0.5 second of each regime is used as partial annotation, according to all metrics. Using only 0.1 second results in a F1 SCORE equal to 0.87, its worst value. Too little information is present in the annotation and the resulting metric is not adapted to the change points to detect. When partial annotations go from 0.5 to 1 or 1.5 seconds, HAUSDORFF increases by more than 0.4 second. This fact can be explained by looking at the segmentation accuracy by change point type, reported in Table 8.5.

♡gkCPD	0.1 sec	0.5 sec	1 sec	1.5 sec
Stand/Walk	0.69 ($\pm$0.88)	0.46 ($\pm$0.39)	0.57 ($\pm$1.03)	0.54 ($\pm$1.01)
Walk/Turnaround	0.71 ($\pm$1.41)	0.37 ($\pm$0.49)	0.90 ($\pm$1.82)	0.81 ($\pm$1.74)
Turnaround/Walk	0.96 ($\pm$1.84)	0.73 ($\pm$1.46)	1.26 ($\pm$2.20)	1.28 ($\pm$2.36)
Walk/Stand	0.70 ($\pm$0.85)	0.54 ($\pm$0.39)	0.47 ($\pm$0.53)	0.53 ($\pm$0.75)

Table 8.5: Accuracy (means and standard deviations) in seconds against the length of labelled data.

When partial annotations have a duration of 1 or 1.5 seconds, the least precisely detected change point is the third one, "Turnaround/Walk". The reason is that samples at the edges of the "Turnaround" regime are included in the training set annotations, even though they are less representative of the underlying model of the regime (they are at the limit between two regimes). By trying to the satisfy the similarity/dissimilarity constraints resulting from those samples, the learned metric is less able to correctly identify the change points. Overall, this implies that using around 50 samples per regime is enough to learn a discriminative metric on the *Gait* data set. This quantity seems to be a characteristic time of our learning strategy, related to the frequency of the observed phenomena, namely the footsteps, whose frequency is usually just below 2 Hz.

4 Discussion: double labels

We illustrate the influence of input annotations on our supervised change point detection procedure. To that end, we create a signal with two types of change points and feed as annotation two different ground truths to learn a metric. The metric is expected to adapt to the input annotations, even though the train signal is the same. The signal contains two types of change points: low-frequency shifts and high-frequency shifts and is constructed as follows: two signals (sum of sines) are generated with high and low frequency change points and then summed to form the final signal. It is given by

$$y_t = y_t^{\text{low}} + y_t^{\text{high}} \tag{8.10}$$

where y_t^{low} and y_t^{high} are noisy sum-of-sines signals:

$$y_t^{\text{low}} = \sin(2\pi f_1^{\text{low}} t) + \sin(2\pi f_2^{\text{low}} t) + \varepsilon_t \tag{8.11}$$

and

$$y_t^{\text{high}} = \sin(2\pi f_1^{\text{high}} t) + \sin(2\pi f_2^{\text{high}} t) + \varepsilon_t \quad (t = 1, \dots, T), \tag{8.12}$$

where f_1^{low}, f_2^{low}, f_1^{high} and f_2^{high} are frequencies and ε_t is a Gaussian white noise with variance σ^2. Signals are $T = 2000$-sample long. The frequency vectors $[f_1^{\text{low}}, f_2^{\text{low}}]$ alternates from $[0.075, 0.1]$ to $[0.125, 0.1]$ at change indexes $t = 500, 1000, 1500$. The frequency vectors $[f_1^{\text{high}}, f_2^{\text{high}}]$ alternates from $[0.275, 0.3]$ to $[0.325, 0.3]$ at change indexes $t = 670, 1300$. Noise is added for an SNR of 5 Db. A signal realization is displayed on Figure 8.3.

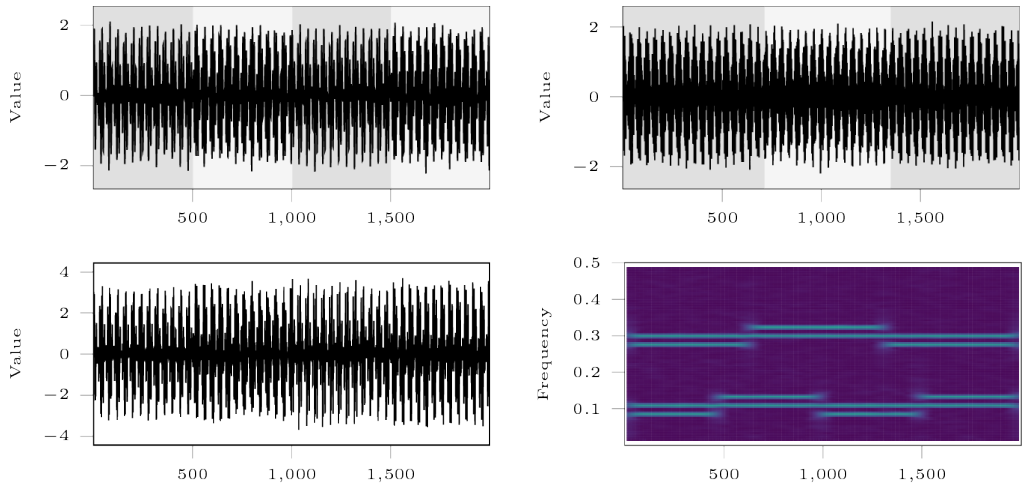


Figure 8.3: Top: signals y^{low} (left) and y^{high} (right). Successive regimes are in alternating grey areas. Bottom: signal y and associated spectrogram.

The metric learning algorithm uses a *linear* kernel and is applied on the time-frequency representation of the signal y. Two metric matrices $\widehat{M}_{\text{high}}$ and $\widehat{M}_{\text{low}}$ are learned by feeding annotations related to the high frequency change points and the low frequency change points, respectively. In both contexts, the train signal is the same. Change points are then estimated, using the greedy method gCPD on a new signal realization using the learned metric.

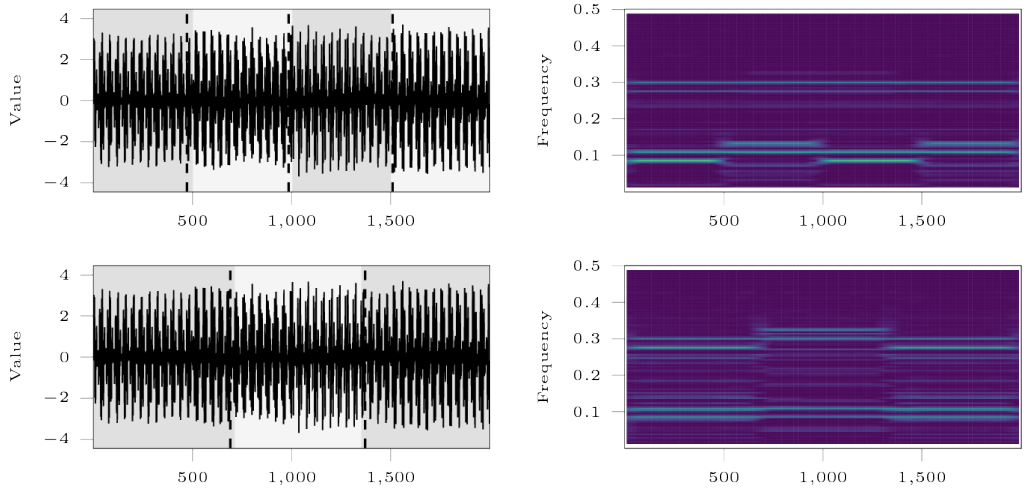


Figure 8.4: Top: signal realization (left) and projected spectrogram (right) for the low frequency change point annotation. Bottom: same signal realization (left) and projected spectrogram (right) for the high frequency change point annotation. Alternating gray areas denote the target segmentation. Dashed lines denote the estimated segmentation.

We verify on Figure 8.4 that the predicted segmentation agrees with the annotation. In other words, detected change points are of the same type (high or low frequency) than the

ones that are given as input annotation. We illustrate how the metric matrix transforms the feature space. Since $\widehat{M}^{\bullet}$ (where $\bullet$ is either "low" or "high") are positive definite matrices, they have a Cholesky decomposition $\widehat{M}^{\bullet} = \widehat{L}^{\bullet}(\widehat{L}^{\bullet})'$. We can see $(\widehat{L}^{\bullet})'$ as a projection matrix. On Figure 8.4 are displayed the projected spectrograms. When they are compared to the unprojected spectrograms (similar to the one displayed on Figure 8.3), we observe that undesirable frequency shifts are toned down by the projection while the "correct" type of change points are preserved. To sum up, our metric learning strategy works as expected and transforms the input space so that undesirable change points are removed.

5 Conclusion

In this chapter, an automatic procedure to calibrate the cost function of change point detection methods has been described. To that end, we show that the calibration problem can be formulated as a kernel metric learning task. Under this setting, a non-parametric transformation of the samples is learned using signal annotations. This data transformation minimizes intra-regime distances and maximizes inter-regime distances. An efficient optimization algorithm is then derived. Our procedure can accommodate full and partial annotations. Once learned, this cost function can be combined with a search method (`Opt`, `Win`, `gkCPD`, etc.) and a complexity constraint (fixed number of changes, linear penalty, `Alpin`, etc.) to create a change point detection algorithm that is able to reproduce the segmentation strategy of the annotations. Numerical experiments show that, for both the linear and Gaussian kernel, supervision improves segmentation accuracy. On the *Gait* data set, our supervised strategy, applied on the raw signals, has the best segmentation accuracy of all methods tested in this thesis. This mitigates the need for a manual design of a suitable signal representation. We also show that this method can be used to recover change points from partially annotated signals, as well as adapt to different labels, even though the signals are the same. An interesting issue for future research is the interpretation the learned metric. For the *Gait* data set, this could give us insights on which parameters are important when detecting changes.

Part IV

Statistical software

9

ruptures: change point detection in Python

Contents

Abstract

ruptures is a Python library for offline change point detection. This package provides methods for the analysis and segmentation of non-stationary signals. Implemented algorithms include exact and approximate detection for various parametric and non-parametric models. ruptures focuses on ease of use by providing a well-documented and consistent interface. In addition, thanks to its modular structure, different algorithms and models can be connected and extended within this package.

1 Introduction

Change point detection is the task of finding changes in the underlying model of a signal. This subject has generated important activity in statistics and signal processing [77, 90, 111]. Modern applications in bioinformatics, finance, monitoring of complex systems have also motivated recent developments from the machine learning community [84, 105, 160].

We present ruptures, a Python scientific library for multiple change point detection in multivariate signals. It is meant to answer the growing need for fast exploration, by non-specialists, of non-stationary signals. In addition, we expect that removing the cost of reimplementation will facilitate composition of new algorithms. To that end, ruptures insists on an easy-to-use and consistent interface. Implementation is also modular to allow users to seamlessly plug their own code.

To the best of the authors' knowledge, ruptures is the first Python package dedicated to multiple change point detection. As displayed in Table 9.1, all related softwares are

Name	Language	Link	Type(s) of change	Algorithm(s)
wbsts [100, 101]	R	cran.r-project.org/package=wbsts	Change in 2nd order stationary structure [125]	BinSeg [67, 100]
trend [135]	R	cran.r-project.org/package=trend	Single shift in trend	Opt
strucchange [168]	R	cran.r-project.org/package=strucchange	Shifts in mean and in linear model	Opt
SeqCBS [148]	R	cran.r-project.org/package=SeqCBS	Mean-shifts in Poisson processes	BinSeg [126]
SegCorr [56]	R	cran.r-project.org/package=SegCorr	Shifts in mean and scale	Opt
cpm [140]	R	cran.r-project.org/package=cpm	Change in Gaussian, exponential, Bernoulli random variables, and general distribution change	Win, BinSeg
not [23]	R	cran.r-project.org/package=not	Mean-shifts in univariate signals (with different types of noise)	BinSeg
factorcpt [49]	R	cran.r-project.org/package=factorcpt	Mean-shifts with factor analysis	BinSeg
ecp [88]	R	cran.r-project.org/package=ecp	Distribution changes	BinSeg, BotUp
changepoint [97]	R	cran.r-project.org/package=changepoint	Mean and scale shifts in univariate signals	Pelt
breakfast [66]	R	cran.r-project.org/package=breakfast	Mean-shifts in univariate signals	BinSeg [63, 65, 67]
bcp [60]	R	cran.r-project.org/package=bcp	Bayesian counterpart of strucchange	Bayesian
changepoint.np [78]	R	cran.r-project.org/package=changepoint.np	Change in distribution (based on empirical distribution function)	Pelt
Segmentor3IsBack [51]	R	cran.r-project.org/package=Segmentor3IsBack	Distribution changes for Gaussian, Poisson, exponential, negative binomial variables	Opt
wbs [22]	R	cran.r-project.org/package=wbs	Mean-shifts in univariate signals	BinSeg [67]
AR1seg [40, 41]	R	cran.r-project.org/package=AR1seg	Mean-shifts in AR(1) processes	Opt

Table 9.1: Summary of available libraries for change point detection.

implemented in R. However, few provide more than one algorithm, and even fewer can be applied to detect changes other than mean-shifts [89, 141]. On the other hand, ruptures contains several standard methods as well as recent contributions, most of which are not available elsewhere (in Python or R). Our work encompasses most packages and provides a unique framework to run and evaluate all algorithms.

In the following, we quickly describe the change point detection framework. Then the main features of the library are detailed.

2 Change point detection framework

In the offline (or retrospective) change point detection framework, we consider a non-stationary random process $y = \{y_1, \dots, y_T\}$ that takes value in $\mathbb{R}^d$ ($d \geq 1$). The signal y is assumed to be piecewise stationary, meaning that some characteristics of the process change abruptly at some unknown instants $t_1^\star < t_2^\star < \cdots < t_K^\star$. Change point detection consists in estimating those instants when a particular realization of y is observed. Note that the number of changes K is not necessarily known.
Most estimation methods adhere to or are an approximation of a general format where a suitable contrast function $C(\cdot)$ is minimized [90, 108]. Usually, it is written as a sum of segment costs:

$$V(\mathcal{T}, y) := c(\{y_t\}_1^{t_1}) + c(\{y_t\}_{t_1+1}^{t_2}) + \cdots + c(\{y_t\}_{t_i+1}^{t_{i+1}}) + \dots \tag{9.1}$$

where $\mathcal{T} = \{t_1, t_2, \dots\}$ denotes a set of change point indexes and $c(\cdot)$ denotes a cost function that takes a process as input and measures its goodness-of-fit to a specified model. The contrast $V(\cdot)$ is the total cost associated with choosing a particular segmentation $\mathcal{T}$. Change point detection amounts to solving the following discrete optimization problem:

$$\min_{\mathcal{T}} V(\mathcal{T}, y) + \text{pen}(\mathcal{T}) \tag{9.2}$$

where $\text{pen}(\mathcal{T})$ is a regularizer on the value of the partition $\mathcal{T}$. Methods from the literature essentially differ by 1) the constraints they add to this optimization problem (fixed dimension of $\mathcal{T}$, penalty term, cost budget, etc.), 2) how they search for the solution (exact or approximate resolution, local or sequential, etc.) and 3) the cost function $c(\cdot)$ they use (which is related to the type of change).

3 Library overview

A basic flowchart is displayed on Figure 9.1. Each block of this diagram is described in the following brief overview of `ruptures`' features. More information can be found in the related documentation (see link to source in Section 3.2).

3.1 Main features

- **Search methods** Our package includes the main algorithms from the literature, namely dynamic programming, detection with a l_0 constraint, binary segmentation, bottom-up segmentation and window-based segmentation. This choice is the result of a trade-off between exhaustiveness and adaptiveness. Rather than providing as many methods as possible, only algorithms which have been used in several different settings are included. In particular, numerous "mean-shift only" detection procedures were not considered. Implemented algorithms have sensible default parameters that can be changed easily through the functions' interface.

- **Cost functions** Cost functions are related to the type of change to detect. Within `ruptures`, one has access to parametric cost functions that can detect shifts in standard statistical quantities (mean, scale, linear relationship between dimensions, autoregressive coefficients, etc.) and non-parametric cost functions (kernel-based or Mahalanobis-type metric) that can, for instance, detect distribution changes [72, 105].

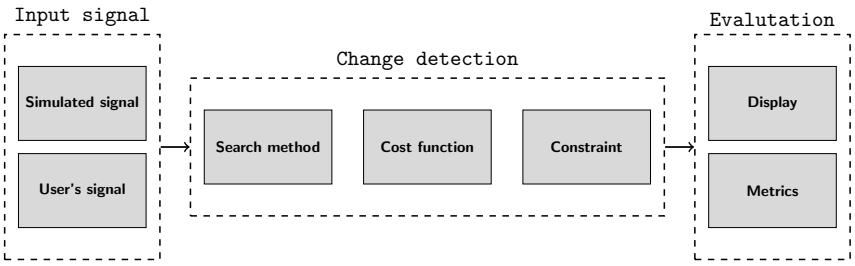


Figure 9.1: Schematic view of the ruptures package.

- **Constraints** All methods can be used whether the number of change points is known or not. In particular, ruptures implements change point detection under a cost budget and with a linear penalty term [98, 121].

- **Evaluation** Evaluation metrics are available to quantitatively compare segmentations, as well as a display module to visually inspect algorithms' performances.

- **Input** Change point detection can be performed on any univariate or multivariate signal that fits into a *Numpy* array. A few standard non-stationary signal generators are included.

- **Consistent interface and modularity** Discrete optimization methods and cost functions are the two main ingredients of change point detection. Practically, each is related to a specific object in the code, making the code highly modular: available optimization methods and cost functions can be connected and composed. An appreciable by-product of this approach is that a new contribution, provided its interface follows a few guidelines, can be integrated seamlessly into ruptures.

- **Scalability** Data exploration often requires to run several times the same methods with different sets of parameters. To that end, a cache is implemented to keep intermediate results in memory, so that the computational cost of running the same algorithm several times on the same signal is greatly reduced. We also add the possibility for a user with speed constraints to sub-sample their signals and set a minimum distance between change points.

3.2 Availability and requirements

The ruptures library is written in pure Python and available on Mac OS X, Linux and Windows platforms. Source code is available from reine.cmla.ens-cachan.fr[1] under the BSD license. We also provide a complete documentation that includes installation instructions, explanations with code snippets on advance use (ctruong.perso.math.cnrs.fr/ruptures). Implementation relies on *Numpy* as the base data structure for signals and parameters and *Scipy* for efficient linear algebra and array operations. The *Matplotlib* library is recommended for visualization. Unit tests (through the *Pytest* library) are provided to facilitate the validation of new pieces of code.

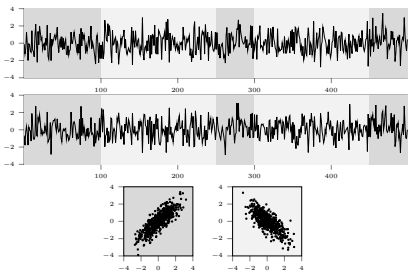
[1] https://reine.cmla.ens-cachan.fr/c.truong/ruptures/repository/latest/archive.zip

```python
import ruptures as rpt

# signal generation
signal, bkps = rpt.pw_normal(n_samples=500, n_bkps=4)

# change point detection
algo = rpt.Dynp(model="rbf").fit(signal)
result = algo.predict(n_bkps=4)
```

(a) Python code.



(b) Top and middle: simulated 2D signal; regimes are highlighted in alternating gray area. Below: scatter plots for each regime type.

Figure 9.2: Illustrative example.

3.3 Illustrative example

As an illustrative example, we perform a kernel change point detection on a simulated piecewise stationary process [72]. In a nutshell, this method maps the input signal onto a high-dimensional Hilbert space $\mathcal{H}$ through a kernel function (here, we use the radial basis function) and searches for mean-shifts.
First, random change point indexes are drawn and a 2D signal of i.i.d. centered normal variables with changing covariance matrix is simulated (Figure 9.2b). The algorithm's internal parameters are then fitted on the data. The discrete minimization of the contrast function is performed with dynamic programming and the associated estimates are returned. The related code lines are reported on Figure 9.2a.
It is worth mentioning that only a few instructions are needed to perform the segmentation. In addition, thanks to `ruptures`, variations of the kernel change point detection can be easily carried out by changing a few parameters in this code.

4 Conclusion

`ruptures` is the most comprehensive change point detection library. Its consistent interface and modularity allow painless comparison between methods and easy integration of new contributions. In addition, a thorough documentation is available for novice users. Thanks to the rich Python ecosystem, `ruptures` can be used in coordination with numerous other scientific libraries

Conclusion and perspectives

In this thesis, several contributions have been proposed for the detection of multiple change points in multivariate signals. The original motivation for this work was the substantial amount of physiological time series collected by monitoring subjects while they undergo a clinical protocol. In this context, change point detection is a critical step in the transition from raw signals to actionable data. To cope with the demanding setting of daily clinical practice, our contributions covered the three elements that characterize detection methods, namely the cost function, the search method and the constraint on the number of changes.

In Part II : Greedy change point detection, a trade-off (in terms of complexity) between exact detection methods and fast window-based methods is described. Precisely, a greedy detection procedure is proposed, that leads to two (related) algorithms: `gCPD` (linear kernel) and `gkCPD` (arbitrary kernel). The first algorithm, `gCPD`, greedily approximates the optimal change point detection solution, using the Orthogonal Matching Pursuit (OMP) strategy. Our algorithm `gCPD` is then extended, leading to `gkCPD`, which is based on a kernel norm. Thanks to the properties of reproducing Hilbert spaces, `gkCPD` detects changes in higher-order moments of probability distributions. Numerical experiments on real-world signals show that both algorithms display competitive results. In particular, greedy approaches are more accurate than standard sub-optimal methods and faster than optimal methods. An interesting direction of research would be to obtain consistency results for the greedy algorithm `gkCPD`, with an arbitrary kernel. A possible approach would be to use concentration inequalities adapted to a non-Gaussian Hilbertian setting, as in [69].
Part III : Supervised change point detection focuses on automatic calibration of detection methods. To that end, two procedures based on supervised learning are described. In both situations, an expert specifies to the algorithm what is considered a change by providing annotated signals. In practice, such annotations can either be "full", when the exact change point locations are provided, or "partial", when only approximate locations are provided. The algorithm then calibrates the detection method so that it replicates the segmentation strategy of the expert. This setting removes the need for a manual tuning of the parameters of the detection method. In addition, several experts that are interested in different phenomena can easily provide different annotations to have a calibration adapted to their needs. The first procedure, `Alpin`, selects the correct number of change points in signals, by learning the smoothing parameter of a linear penalty, using a training set of fully annotated signals. Precisely, `Alpin` consists in minimizing a particular loss function, the excess risk. Numerical experiments on synthetic and real-world data sets show that `Alpin` outperforms standard non-supervised penalization methods. In addition, compared to other supervised approaches, `Alpin` is faster and can be applied to detect arbitrary types of changes. In the future, it would be interesting to extend `Alpin` to partial labels. Also, as done in [82], it is possible to have the smoothing parameter depend on key parameters of the signal (length, noise level, etc.) through a linear relationship. An easily modified version of `Alpin` could be developed and tested on real-world data. The second contribution of Part III deals with the calibration of the cost function. Alternatively, this

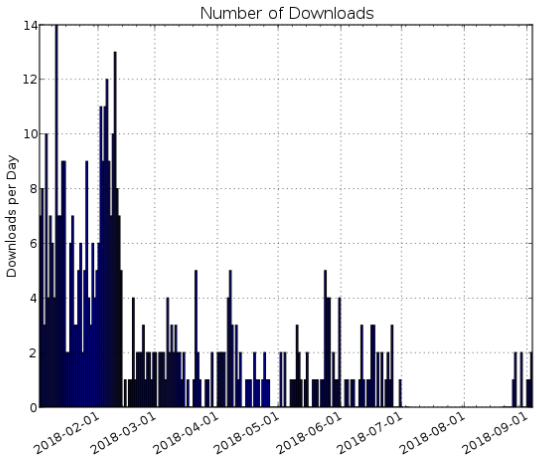


Figure 9.3: Number of downloads of ruptures per day (as of 9/5/2018) from the Machine Learning Open Source Software (MLOSS) platform[2]. (This is only one of the platforms on which ruptures is available.)

can be seen as finding a signal representation so that annotated changes, however complex, are transformed into mean-shifts, which are a well-studied type of change. This calibration problem is formulated as a kernel metric learning task. Our procedure can accommodate full and partial annotations. Numerical experiments show that, for both the linear and Gaussian kernel, supervision improves segmentation accuracy. On the *Gait* data set, our supervised strategy, applied on the raw signals, has the best segmentation accuracy of all methods tested in this thesis. This mitigates the need for a manual design of a suitable signal representation. We also show that this method can be used to recover change points from partially annotated signals, as well as adapt to different labels, even though the signals are the same. A promising perspective would be to generalize the methodology of Part III in order to adapt other supervised learning procedures to change point detection. At the time of writing, only a few works use full or partial signal annotations to learn from an expert a segmentation strategy. In our opinion, such an approach is likely to have considerable practical and theoretical consequences for change point detection.

To facilitate the use of segmentation methods, implementations of standard algorithms can be found in the Python package ruptures, which is described in Part IV : Statistical software. ruptures is the most comprehensive library in Python. Its consistent interface and modularity allow painless comparison between methods and easy integration of new contributions. In addition, a thorough documentation is available for novice users. Thanks to the rich Python ecosystem, ruptures can be used in coordination with numerous other scientific libraries. Since its release, several contributors have participated in its development, especially through the Github platform. Figure 9.3 shows the number of downloads since the release of ruptures. An immediate addition to ruptures would be our supervised contributions (from Part III).

[2]Several change points are visible on Figure 9.3. From release day to mid-February, ruptures is frequently downloaded because the link to the package is on the front page and a lot of internet bots are scraping the MLOSS website at each new post. From mid-February to July, the number of downloads is relatively constant. Activity completely stops during the summer break to resume in September.

On a concluding note, we would like to point out that the challenges that motivated this thesis can be found in numerous other settings: as more and more systems and individuals are monitored, change point detection has become more and more crucial to contextualize the long time series that are collected. Without a doubt, the methodology and solutions described in this work have the potential to be successfully used in countless other situations, such as self-quantified health, industrial system surveillance, quality control, etc.

A

Documentation of ruptures

ruptures Documentation

Release

Charles Truong

Apr 09, 2018

CONTENTS

`ruptures` is designed to perform offline change point algorithms within the Python language. Also in this library, new methods are presented.

Note:

Listing 1: Basic usage

```
import matplotlib.pyplot as plt
import ruptures as rpt
# generate signal
n_samples, dim, sigma = 1000, 3, 4
n_bkps = 4  # number of breakpoints
signal, bkps = rpt.pw_constant(n_samples, dim, n_bkps, noise_std=sigma)
# detection
algo = rpt.Pelt(model="rbf").fit(signal)
result = algo.predict(pen=10)
# display
rpt.display(signal, bkps, result)
plt.show()
```

CHAPTER

ONE

GETTING STARTED

1.1 License

This project is under BSD license.

1.2 Installation

With pip3 from terminal: `$ pip3 install ruptures`.

Or download the source codes from latest release and run the following lines from inside the folder `$ python3 setup.py install` or `$ python3 setup.py develop`.

1.3 User guide

This section explains how to use implemented algorithms. `ruptures` has an object-oriented modelling approach: change point detection algorithms are broken down into two conceptual objects that inherits from base classes: `BaseEstimator` and `BaseCost`.

1.3.1 Initializing a new estimator

Each change point detection algorithm inherits from the base class `ruptures.base.BaseEstimator`. When a class that inherits from the base estimator is created, the `.__init__()` method initializes an estimator with the following arguments:

- `'model'`: “l1”, “l2”, “normal”, “rbf”, “linear”, “ar”. Cost function to use to compute the approximation error.
- `'cost'`: a custom cost function to the detection algorithm. Should be a `BaseCost` instance.
- `'jump'`: reduce the set of possible change point indexes; predicted change points can only be a multiple of `'jump'`.
- `'min_size'`: minimum number of samples between two change points.

1.3.2 Making a prediction

The main methods are `.fit()`, `.predict()`, `.fit_predict()`:

- `.fit()`: generally takes a signal as input and fit the algorithm on the data

- `.predict()`: performs the change point detection. This method returns a list of indexes corresponding to the end of each regimes. By design, the last element of this list is the number of samples.
- `.fit_predict()`: helper method which calls `.fit()` and `.predict()` successively.

1.3.3 Creating a new cost function

In order to define custom cost functions, simply create a class that inherits from `ruptures.base.BaseCost` and implement the methods `.fit(signal)` and `.error(start, end)`:

- The method `.fit(signal)` takes a signal as input and sets parameters. It returns `'self'`.
- The method `.error(start, end)` takes two indexes `'start'` and `'end'` and returns the cost on the segment start:end.

An example can be found in *Custom cost class*.

CHAPTER

TWO

DOCUMENTATION

The complete documentation can be found here.

2.1 Change point detection: a general formulation

A general framework is introduced in this review of methods *[CTOV18]*.

References

2.2 Search methods

The *ruptures.detection* module implements the change point detection methods.

2.2.1 Exact segmentation: dynamic programming

Description

The method is implemented in *ruptures.detection.Dynp*.

Roughly speaking, it computes the cost of all subsequences of a given signal. The number of computed costs is of the order $\mathcal{O}(Kn^2)$, where K is the number of change points and n the number of samples. This has to be multiplied by the computational cost of computing the approximation error on one sub-sequence. Consequently, piecewise constant models are significantly faster than linear or autoregressive models.

Computational cost is drastically reduced when considering only a subsample of possible change points. When calling *ruptures.detection.Dynp.__init__()*, the minimum distance between change points can be set through the keyword `'min_size'`; through the parameter `'jump'`, only change point indexes multiple of a particular value are considered.

Usage

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt

# creation of data
n, dim = 500, 3
n_bkps, sigma = 3, 5
```

```
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)

# change point detection
model = "l1"  # "l2", "rbf"
algo = rpt.Dynp(model=model, min_size=3, jump=5).fit(signal)
my_bkps = algo.predict(n_bkps=3)

# show results
rpt.show.display(signal, bkps, my_bkps, figsize=(10, 6))
plt.show()
```

Code explanation

class ruptures.detection.**Dynp**(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Find optimal change points using dynamic programming.

Given a segment model, it computes the best partition for which the sum of errors is minimum.

__init__(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Creates a Dynp instance.

Parameters

- **model** (*str, optional*) – segment model, ["l1", "l2", "rbf"]. Not used if `'custom_cost'` is not None.
- **custom_cost** (*BaseCost, optional*) – custom cost function. Defaults to None.
- **min_size** (*int, optional*) – minimum segment length.
- **jump** (*int, optional*) – subsample (one every *jump* points).
- **params** (*dict, optional*) – a dictionary of parameters for the cost instance.

Returns self

fit(*signal*)

Create the cache associated with the signal.

Dynamic programming is a recurrence; intermediate results are cached to speed up computations. This method sets up the cache.

Parameters **signal** (*array*) – signal. Shape (n_samples, n_features) or (n_samples,).

Returns self

fit_predict(*signal*, *n_bkps*)

Fit to the signal and return the optimal breakpoints.

Helper method to call fit and predict once

Parameters

- **signal** (*array*) – signal. Shape (n_samples, n_features) or (n_samples,).
- **n_bkps** (*int*) – number of breakpoints.

Returns sorted list of breakpoints

Return type list

predict(*n_bkps*)

Return the optimal breakpoints.

Must be called after the fit method. The breakpoints are associated with the signal passed to fit().

Parameters **n_bkps** (*int*) – number of breakpoints.

Returns sorted list of breakpoints

Return type list

2.2.2 Exact segmentation: Pelt

Description

The method is implemented in *ruptures.detection.Pelt*.

Because the enumeration of all possible partitions impossible, the algorithm relies on a pruning rule. Many indexes are discarded, greatly reducing the computational cost while retaining the ability to find the optimal segmentation. The implementation follows *[BKFE12]*. In addition, under certain conditions on the change point repartition, the computational complexity is linear on average.

When calling *ruptures.detection.Pelt.__init__()*, the minimum distance between change points can be set through the keyword `'min_size'`; through the parameter `'jump'`, only change point indexes multiple of a particular value are considered.

Usage

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt

# creation of data
n, dim = 500, 3
n_bkps, sigma = 3, 1
signal, b = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)

# change point detection
model = "l1"  # "l2", "rbf"
algo = rpt.Pelt(model=model, min_size=3, jump=5).fit(signal)
my_bkps = algo.predict(pen=3)

# show results
fig, (ax,) = rpt.display(signal, bkps, my_bkps, figsize=(10, 6))
plt.show()
```

Code explanation

class ruptures.detection.**Pelt**(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Penalized change point detection.

For a given model and penalty level, computes the segmentation which minimizes the constrained sum of approximation errors.

__init__ (*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)
Initialize a Pelt instance.

Parameters

- **model** (*str, optional*) – segment model, ["l1", "l2", "rbf"]. Not used if 'custom_cost' is not None.
- **custom_cost** (*BaseCost, optional*) – custom cost function. Defaults to None.
- **min_size** (*int, optional*) – minimum segment length.
- **jump** (*int, optional*) – subsample (one every *jump* points).
- **params** (*dict, optional*) – a dictionary of parameters for the cost instance.

Returns self

fit (*signal*)
Set params.

Parameters **signal** (*array*) – signal to segment. Shape (n_samples, n_features) or (n_samples,).

Returns self

fit_predict (*signal*, *pen*)
Fit to the signal and return the optimal breakpoints.

Helper method to call fit and predict once

Parameters

- **signal** (*array*) – signal. Shape (n_samples, n_features) or (n_samples,).
- **pen** (*float*) – penalty value (>0)

Returns sorted list of breakpoints

Return type list

predict (*pen*)
Return the optimal breakpoints.

Must be called after the fit method. The breakpoints are associated with the signal passed to fit().

Parameters **pen** (*float*) – penalty value (>0)

Returns sorted list of breakpoints

Return type list

References

2.2.3 Binary segmentation

Description

Binary change point detection is used to perform fast signal segmentation and is implemented in `ruptures.detection.BinSeg`. It is a sequential approach: first, one change point is detected in the complete input signal, then series is split around this change point, then the operation is repeated on the two resulting sub-signals. See for instance *[BSBai97]* and *[BSFry14]* for a theoretical and algorithmic analysis of `ruptures.detection.BinSeg`. The benefits of binary segmentation includes low complexity (of the order of $\mathcal{O}(n \log n)$, where n is the number of

samples), the fact that it can extend any single change point detection method to detect multiple changes points and that it can work whether the number of regimes is known beforehand or not.

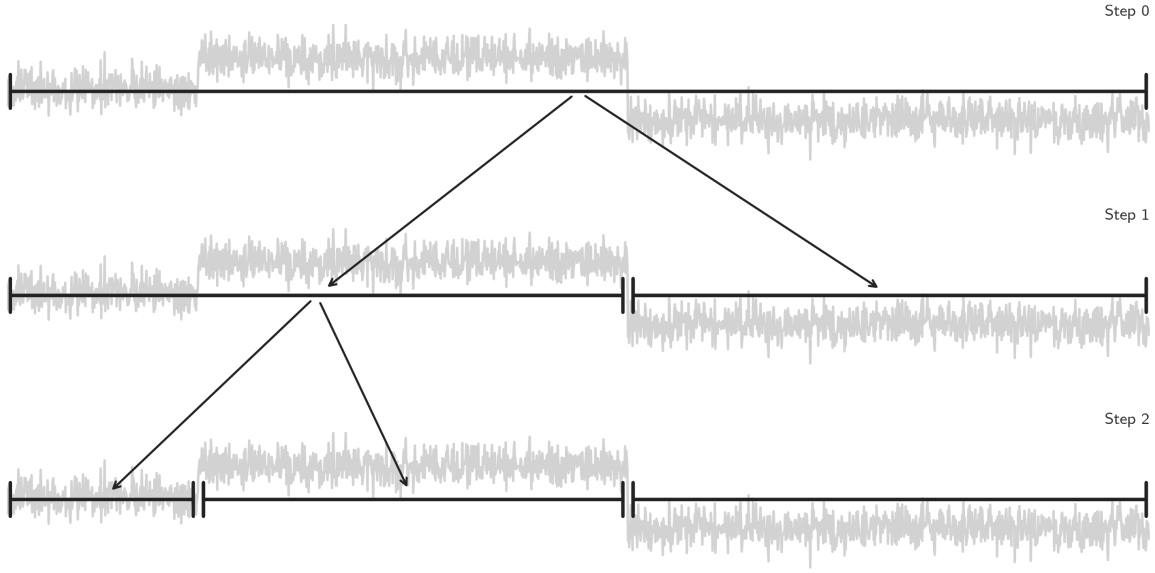


Fig. 2.1: Schematic view of the binary segmentation algorithm.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n = 500  # number of samples
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

To perform a binary segmentation of a signal, initialize a `ruptures.detection.BinSeg` instance.

```
# change point detection
model = "l2"  # "l1", "rbf", "linear", "normal", "ar"
algo = rpt.Binseg(model=model).fit(signal)
my_bkps = algo.predict(n_bkps=3)

# show results
rpt.show.display(signal, bkps, my_bkps, figsize=(10, 6))
plt.show()
```

In the situation in which the number of change points is unknown, one can specify a penalty using the `'pen'` parameter or a threshold on the residual norm using `'epsilon'`.

```
my_bkps = algo.predict(pen=np.log(n)*dim*sigma**2)
# or
my_bkps = algo.predict(epsilon=3*n*sigma**2)
```

See also:

Change point detection: a general formulation for more information about stopping rules of sequential algorithms.

For faster predictions, one can modify the `'jump'` parameter during initialization. The higher it is, the faster the prediction is achieved (at the expense of precision).

```
algo = rpt.Binseg(model=model, jump=10).fit(signal)
```

Code explanation

class ruptures.detection.**Binseg**(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Binary segmentation.

__init__(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Initialize a Binseg instance.

Parameters

- **model** (*str, optional*) – segment model, ["l1", "l2", "rbf",...]. Not used if `'custom_cost'` is not None.
- **custom_cost** (*BaseCost, optional*) – custom cost function. Defaults to None.
- **min_size** (*int, optional*) – minimum segment length. Defaults to 2 samples.
- **jump** (*int, optional*) – subsample (one every *jump* points). Defaults to 5 samples.
- **params** (*dict, optional*) – a dictionary of parameters for the cost instance.

Returns self

fit(*signal*)

Compute params to segment signal.

Parameters **signal** (*array*) – signal to segment. Shape (n_samples, n_features) or (n_samples,).

Returns self

fit_predict(*signal*, *n_bkps=None*, *pen=None*, *epsilon=None*)

Fit to the signal and return the optimal breakpoints.

Helper method to call fit and predict once

Parameters

- **signal** (*array*) – signal. Shape (n_samples, n_features) or (n_samples,).
- **n_bkps** (*int*) – number of breakpoints.
- **penalty** (*float*) – penalty value (>0)
- **epsilon** (*float*) – reconstruction budget (>0)

Returns sorted list of breakpoints

Return type list

predict(*n_bkps=None*, *pen=None*, *epsilon=None*)

Return the optimal breakpoints.

Must be called after the fit method. The breakpoints are associated with the signal passed to fit(). The stopping rule depends on the parameter passed to the function.

Parameters

- **n_bkps** (*int*) – number of breakpoints to find before stopping.
- **penalty** (*float*) – penalty value (>0)
- **epsilon** (*float*) – reconstruction budget (>0)

Returns sorted list of breakpoints

Return type list

References

2.2.4 Bottom-up segmentation

Description

Bottom-up change point detection is used to perform fast signal segmentation and is implemented in *ruptures.detection.BottomUp*. It is a sequential approach. Contrary to binary segmentation, which is a greedy procedure, bottom-up segmentation is generous: it starts with many change points and successively deletes the less significant ones. First, the signal is divided in many sub-signals along a regular grid. Then contiguous segments are successively merged according to a measure of how similar they are. See for instance *[BUKCHP01]* or *[BUFry07]* for an algorithmic analysis of *ruptures.detection.BottomUp*. The benefits of bottom-up segmentation includes low complexity (of the order of $\mathcal{O}(n \log n)$, where n is the number of samples), the fact that it can extend any single change point detection method to detect multiple changes points and that it can work whether the number of regimes is known beforehand or not.

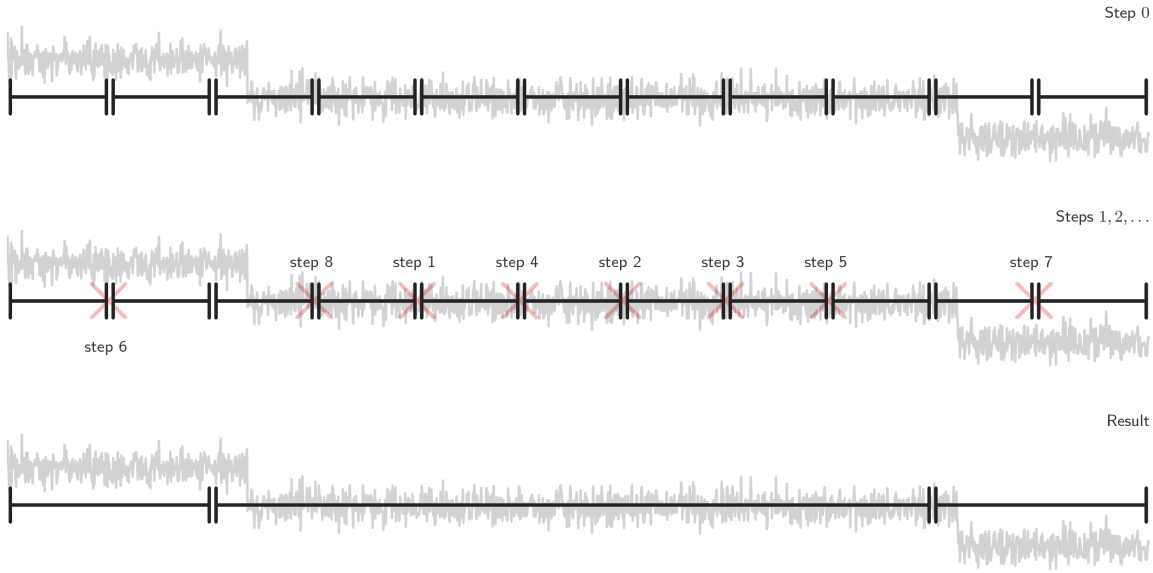


Fig. 2.2: Schematic view of the bottom-up segmentation algorithm.

See also:

Binary segmentation.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

To perform a bottom-up segmentation of a signal, initialize a *ruptures.detection.BottomUp* instance.

```
# change point detection
model = "l2"  # "l1", "rbf", "linear", "normal", "ar"
algo = rpt.BottomUp(model=model).fit(signal)
my_bkps = algo.predict(n_bkps=3)

# show results
rpt.show.display(signal, bkps, my_bkps, figsize=(10, 6))
plt.show()
```

In the situation in which the number of change points is unknown, one can specify a penalty using the `'pen'` parameter or a threshold on the residual norm using `'epsilon'`.

```
my_bkps = algo.predict(pen=np.log(n)*dim*sigma**2)
# or
my_bkps = algo.predict(epsilon=3*n*sigma**2)
```

See also:

Change point detection: a general formulation for more information about stopping rules of sequential algorithms.

For faster predictions, one can modify the `'jump'` parameter during initialization. The higher it is, the faster the prediction is achieved (at the expense of precision).

```
algo = rpt.BottomUp(model=model, jump=10).fit(signal)
```

Code explanation

class ruptures.detection.**BottomUp**(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Bottom-up segmentation.

__init__(*model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Initialize a BottomUp instance.

Parameters

- **model** (*str, optional*) – segment model, ["l1", "l2", "rbf"]. Not used if `'custom_cost'` is not None.
- **custom_cost** (*BaseCost, optional*) – custom cost function. Defaults to None.
- **min_size** (*int, optional*) – minimum segment length. Defaults to 2 samples.
- **jump** (*int, optional*) – subsample (one every *jump* points). Defaults to 5 samples.
- **params** (*dict, optional*) – a dictionary of parameters for the cost instance.

Returns self

fit (*signal*)
Compute params to segment signal.

Parameters **signal** (*array*) – signal to segment. Shape (n_samples, n_features) or (n_samples,).

Returns self

fit_predict (*signal*, *n_bkps=None*, *pen=None*, *epsilon=None*)
Fit to the signal and return the optimal breakpoints.

Helper method to call fit and predict once

Parameters

- **signal** (*array*) – signal. Shape (n_samples, n_features) or (n_samples,).
- **n_bkps** (*int*) – number of breakpoints.
- **penalty** (*float*) – penalty value (>0)
- **epsilon** (*float*) – reconstruction budget (>0)

Returns sorted list of breakpoints

Return type list

predict (*n_bkps=None*, *pen=None*, *epsilon=None*)
Return the optimal breakpoints.

Must be called after the fit method. The breakpoints are associated with the signal passed to fit(). The stopping rule depends on the parameter passed to the function.

Parameters

- **n_bkps** (*int*) – number of breakpoints to find before stopping.
- **penalty** (*float*) – penalty value (>0)
- **epsilon** (*float*) – reconstruction budget (>0)

Returns sorted list of breakpoints

Return type list

References

2.2.5 Window-based change point detection

Description

Window-based change point detection is used to perform fast signal segmentation and is implemented in *ruptures.detection.Window*. The algorithm uses two windows which slide along the data stream. The statistical properties of the signals within each window are compared with a discrepancy measure. For a given cost function $c(\cdot)$ (see *Cost functions*), a discrepancy measure is derived $d(\cdot, \cdot)$ as follows:

$$d(y_{u..v}, y_{v..w}) = c(y_{u..w}) - c(y_{u..v}) - c(y_{v..w})$$

where $\{y_t\}_t$ is the input signal and $u < v < w$ are indexes. The discrepancy is the cost gain of splitting the sub-signal $y_{u..w}$ at the index v. If the sliding windows $u..v$ and $v..w$ both fall into a segment, their statistical properties are similar and the discrepancy between the first window and the second window is low. If the sliding windows fall into

two dissimilar segments, the discrepancy is significantly higher, suggesting that the boundary between windows is a change point. The discrepancy curve is the curve, defined for all indexes t between $w/2$ and $n - w/2$ (n is the number of samples),

$$\big(t, d(y_{t-w/2..t}, y_{t..t+w/2})\big)$$

where w is the window length. A sequential peak search is performed on the discrepancy curve in order to detect change points.

The benefits of window-based segmentation includes low complexity (of the order of $\mathcal{O}(nw)$, where n is the number of samples), the fact that it can extend any single change point detection method to detect multiple changes points and that it can work whether the number of regimes is known beforehand or not.

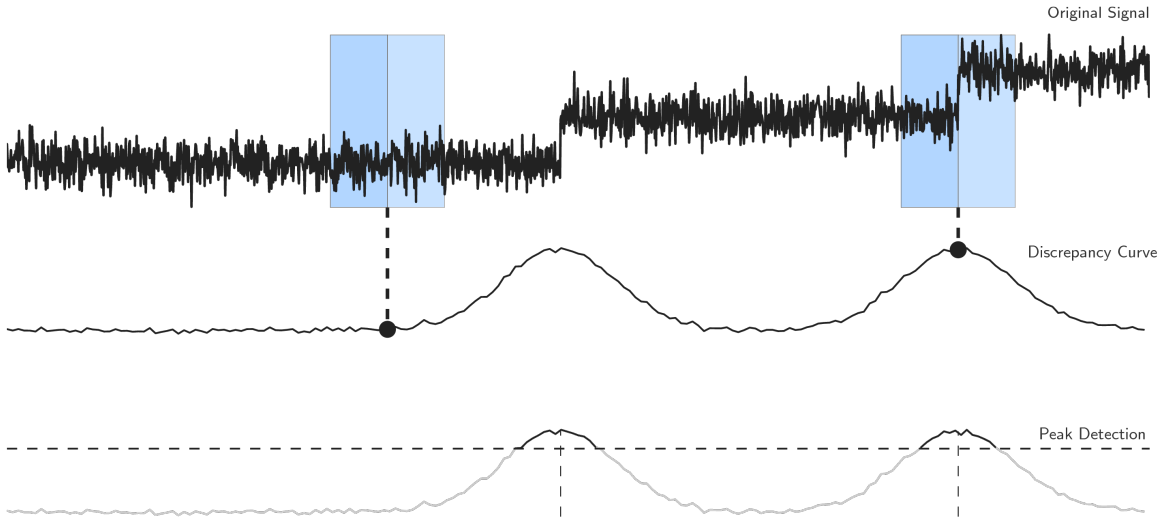


Fig. 2.3: Schematic view of the window sliding algorithm.

See also:

Binary segmentation, *Bottom-up segmentation*.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

To perform a binary segmentation of a signal, initialize a *ruptures.detection.Window* instance.

```
# change point detection
model = "l2"  # "l1", "rbf", "linear", "normal", "ar"
```

```
algo = rpt.Window(width=40, model=model).fit(signal)
my_bkps = algo.predict(n_bkps=3)

# show results
rpt.show.display(signal, bkps, my_bkps, figsize=(10, 6))
plt.show()
```

The window length (in number of samples) is modified through the argument `'width'`. Usual methods assume that the window length is smaller than the smallest regime length.

In the situation in which the number of change points is unknown, one can specify a penalty using the `'pen'` parameter or a threshold on the residual norm using `'epsilon'`.

```
my_bkps = algo.predict(pen=np.log(n)*dim*sigma**2)
# or
my_bkps = algo.predict(epsilon=3*n*sigma**2)
```

See also:

Change point detection: a general formulation for more information about stopping rules of sequential algorithms.

For faster predictions, one can modify the `'jump'` parameter during initialization. The higher it is, the faster the prediction is achieved (at the expense of precision).

```
algo = rpt.Window(model=model, jump=10).fit(signal)
```

Code explanation

class ruptures.detection.**Window**(*width=100*, *model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Window sliding method.

__init__(*width=100*, *model='l2'*, *custom_cost=None*, *min_size=2*, *jump=5*, *params=None*)

Instanciate with window length.

Parameters

- **width** (*int, optional*) – window length. Defaults to 100 samples.
- **model** (*str, optional*) – segment model, ["l1", "l2", "rbf"]. Not used if
- **is not None.** (*'custom_cost'*) –
- **custom_cost** (*BaseCost, optional*) – custom cost function. Defaults to None.
- **min_size** (*int, optional*) – minimum segment length.
- **jump** (*int, optional*) – subsample (one every *jump* points).
- **params** (*dict, optional*) – a dictionary of parameters for the cost instance.

Returns self

fit(*signal*)

Compute params to segment signal.

Parameters **signal** (*array*) – signal to segment. Shape (n_samples, n_features) or (n_samples,).

Returns self

fit_predict (*signal*, *n_bkps=None*, *pen=None*, *epsilon=None*)
Helper method to call fit and predict once.

predict (*n_bkps=None*, *pen=None*, *epsilon=None*)
Return the optimal breakpoints.

Must be called after the fit method. The breakpoints are associated with the signal passed to fit(). The stopping rule depends on the parameter passed to the function.

Parameters

- **n_bkps** (*int*) – number of breakpoints to find before stopping.
- **penalty** (*float*) – penalty value (>0)
- **penalty** – penalty value

Returns sorted list of breakpoints

Return type list

2.3 Cost functions

2.3.1 Least absolute deviation

Description

This cost function detects changes in the median of a signal. Overall, it is a robust estimator of a shift in the central point (mean, median, mode) of a distribution *[C1Bai95]*. Formally, for a signal $\{y_t\}_t$ on an interval I,

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

where $\bar{y}$ is the componentwise median of $\{y_t\}_{t \in I}$.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

Then create a `CostL1` instance and print the cost of the sub-signal `signal[50:150]`.

```
c = rpt.costs.CostL1().fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostL1` instance (through the argument `'custom_cost'`) or set `model="l1"`.

```
c = rpt.costs.CostL1(); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="l1")
```

Code explanation

class ruptures.costs.**CostL1**

Least absolute deviation.

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit(*signal*)

Set parameters of the instance.

Parameters **signal** (*array*) – signal. Shape (n_samples,) or (n_samples, n_features)

Returns self

References

2.3.2 Least squared deviation

Description

This cost function detects mean-shifts in a signal. Formally, for a signal $\{y_t\}_t$ on an interval I,

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

where $\bar{y}$ is the mean of $\{y_t\}_{t \in I}$.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

Then create a `CostL2` instance and print the cost of the sub-signal `signal[50:150]`.

```
c = rpt.costs.CostL2().fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostL2` instance (through the argument `'custom_cost'`) or set `model="l2"`.

```
c = rpt.costs.CostL2(); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="l2")
```

Code explanation

class ruptures.costs.**CostL2**

Least squared deviation.

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit(*signal*)

Set parameters of the instance.

Parameters **signal** (*array*) – signal. Shape (n_samples,) or (n_samples, n_features)

Returns self

2.3.3 Gaussian process change

Description

This cost function detects changes in the mean and scale of a Gaussian time series. Formally, for a signal $\{y_t\}_t$ on an interval I,

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

where $\widehat{\Sigma}_I$ is the empirical covariance matrix of the sub-signal $\{y_t\}_{t\in I}$.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

Then create a `CostNormal` instance and print the cost of the sub-signal `signal[50:150]`.

```
c = rpt.costs.CostNormal().fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostNormal` instance (through the argument `'custom_cost'`) or set `model="normal"`.

```
c = rpt.costs.CostNormal(); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="normal")
```

Code explanation

class ruptures.costs.**CostNormal**

Maximum Gaussian likelihood.

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit (*signal*)
Set parameters of the instance.

Parameters **signal** (*array*) – signal. Shape (n_samples,) or (n_samples, n_features)

Returns self

2.3.4 Kernelized mean change

Description

Given a positive semi-definite kernel $k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ and its associated feature map $\Phi : \mathbb{R}^d \mapsto \mathcal{H}$ (where $\mathcal{H}$ is an appropriate Hilbert space), this cost function detects changes in the mean of the embedded signal $\{\Phi(y_t)\}_t$ *[KERACH12][KERGBR+12]*. Formally, for a signal $\{y_t\}_t$ on an interval I,

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

where $\bar{\mu}$ is the empirical mean of the embedded sub-signal $\{\Phi(y_t)\}_{t \in I}$. Here the kernel is the radial basis function (rbf):

$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$

where $\|\cdot\|$ is the Euclidean norm and $\gamma > 0$ is the so-called bandwidth parameter and is determined according to median heuristics (i.e. equal to the inverse of median of all pairwise distances).

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

Then create a `CostRbf` instance and print the cost of the sub-signal `signal[50:150]`.

```
c = rpt.costs.CostRbf().fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostRbf` instance (through the argument `'custom_cost'`) or set `model="rbf"`.

```
c = rpt.costs.CostRbf(); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="rbf")
```

Code explanation

class ruptures.costs.**CostRbf**

Kernel cost function (rbf kernel).

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit(*signal*)

Sets parameters of the instance.

Parameters **signal** (*array*) – signal. Shape (n_samples,) or (n_samples, n_features)

Returns self

References

2.3.5 Linear model change

Description

Let $0 < t_1 < t_2 < \cdots < n$ be unknown change points indexes. Consider the following multiple linear regression model

$$y_t = z_t' \delta_j + \varepsilon_t, \quad \forall t = t_j, \ldots, t_{j+1} - 1$$

for $j > 1$. Here, the observed dependant variable is $y_t \in \mathbb{R}$, the covariate vector is $x_t \in \mathbb{R}^p$, the disturbance is $\varepsilon_t \in \mathbb{R}$. The vectors $\delta_j \in \mathbb{R}^p$ are the paramater vectors (or regression coefficients).

The least-squares estimates of the break dates is obtained by minimiming the sum of squared residuals *[CLBP03]*. Formally, the associated cost function on an interval I is

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

Usage

Start with the usual imports and create a signal with piecewise linear trends.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, n_reg = 2000, 3  # number of samples, number of regressors (including intercept)
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
# regressors
tt = np.linspace(0, 10*np.pi, n)
X = np.vstack((np.sin(tt), np.sin(5*tt), np.ones(n))).T
# parameter vectors
deltas, bkps = rpt.pw_constant(n, n_reg, n_bkps, noise_std=None, delta=(1, 3))
# observed signal
y = np.sum(X*deltas, axis=1)
y += np.random.normal(size=y.shape)
# display signal
rpt.show.display(y, bkps, figsize=(10, 6))
plt.show()
```

Then create a `CostLinear` instance and print the cost of the sub-signal `signal[50:150]`.

```
# stack observed signal and regressors.
# first dimension is the observed signal.
signal = np.column_stack((y.reshape(-1, 1), X))
c = rpt.costs.CostLinear().fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostLinear` instance (through the argument `'custom_cost'`) or set `model="linear"`.

```
c = rpt.costs.CostLinear(); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="linear")
```

Code explanation

class ruptures.costs.**CostLinear**

Least-squares estimate for linear changes.

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit (*signal*)
Set parameters of the instance. The first column contains the observed variable. The other columns contains the covariates.

Parameters **signal** (*array*) – signal. Shape (n_samples, n_regressors+1)

Returns self

References

2.3.6 Autoregressive model change

Description

Let $0 < t_1 < t_2 < \cdots < n$ be unknown change points indexes. Consider the following piecewise autoregressive model

$$y_t = z_t' \delta_j + \varepsilon_t, \quad \forall t = t_j, \ldots, t_{j+1} - 1$$

where $j > 1$ is the segment number, $z_t = [y_{t-1}, y_{t-2}, \ldots, y_{t-p}]$ is the lag vector,and $p > 0$ is the order of the process.

The least-squares estimates of the break dates is obtained by minimiming the sum of squared residuals *[ARBai00]*. Formally, the associated cost function on an interval I is

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

Usage

Start with the usual imports and create a signal with piecewise linear trends.

```
from itertools import cycle
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n = 2000
n_bkps, sigma = 4, 0.5  # number of change points, noise standart deviation
bkps = [400, 1000, 1300, 1800, n]
f1 = np.array([0.075, 0.1])
f2 = np.array([0.1, 0.125])
freqs = np.zeros((n, 2))
for sub, val in zip(np.split(freqs, bkps[:-1]), cycle([f1, f2])):
    sub += val
tt = np.arange(n)
signal = np.sum((np.sin(2*np.pi*tt*f) for f in freqs.T))
signal += np.random.normal(scale=sigma, size=signal.shape)
# display signal
rpt.show.display(signal, bkps, figsize=(10, 6))
plt.show()
```

Then create a `CostAR` instance and print the cost of the sub-signal `signal[50:150]`. The autoregressive order can be specified through the keyword `'order'`.

```
c = rpt.costs.CostAR(order=10).fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostAR` instance (through the argument `'custom_cost'`) or set `model="ar"`. Additional parameters can be passed to the cost instance through the keyword `'params'`.

```
c = rpt.costs.CostAR(order=10); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="ar", params={"order": 10})
```

Code explanation

class ruptures.costs.**CostAR**(*order=4*)

Least-squares estimate for changes in autoregressive coefficients.

error(*start*, *end*)

Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises `NotEnoughPoints` – when the segment is too short (less than `'min_size'` samples).

fit(*signal*)

Set parameters of the instance. The signal must be 1D.

Parameters **signal** (*array*) – 1d signal. Shape (n_samples, 1) or (n_samples,).

Returns self

References

2.3.7 Mahalanobis-type metric

Description

Given a positive semi-definite matrix $M \in \mathbb{R}^{d \times d}$, this cost function detects changes in the mean of the embedded signal defined by the pseudo-metric

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

Formally, for a signal $\{y_t\}_t$ on an interval I, the cost function is equal to

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

where $\bar{\mu}$ is the empirical mean of the sub-signal $\{y_t\}_{t \in I}$. The matrix M can for instance be the result of a similarity learning algorithm *[MLXJR03]* or the inverse of the empirical covariance matrix (yielding the Mahalanobis distance).

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
```

Then create a `CostMl` instance and print the cost of the sub-signal `signal[50:150]`.

```
M = np.eye(dim)
c = rpt.costs.CostMl(metric=M).fit(signal)
print(c.error(50, 150))
```

You can also compute the sum of costs for a given list of change points.

```
print(c.sum_of_costs(bkps))
print(c.sum_of_costs([10, 100, 200, 250, n]))
```

In order to use this cost class in a change point detection algorithm (inheriting from `BaseEstimator`), either pass a `CostMl` instance (through the argument `'custom_cost'`) or set `model="mahalanobis"`.

```
c = rpt.costs.CostMl(metric=M); algo = rpt.Dynp(custom_cost=c)
# is equivalent to
algo = rpt.Dynp(model="mahalanobis", params={"metric": M})
```

Code explanation

class ruptures.costs.**CostMl**(*metric=None*)

Mahalanobis-type cost function.

__init__(*metric=None*)

Create a new instance.

Parameters **metric** (*ndarray, optional*) – PSD matrix that defines a Mahalanobis-type pseudo distance. If None, defaults to the Mahalanobis matrix. Shape (n_features, n_features).

Returns self

error(*start*, *end*)
Return the approximation cost on the segment [start:end].

Parameters

- **start** (*int*) – start of the segment
- **end** (*int*) – end of the segment

Returns segment cost

Return type float

Raises NotEnoughPoints – when the segment is too short (less than 'min_size' samples).

fit(*signal*)
Sets parameters of the instance.

Parameters **signal** (*array*) – signal. Shape (n_samples,) or (n_samples, n_features)

Returns self

References

2.3.8 Custom cost class

Users who are interested in detecting a specific type of change can easily do so by creating a custom cost function. Provided, they use the base cost function `ruptures.base.BaseCost`, they will be able to seamlessly run the algorithms implemented in `ruptures`.

See also:

Creating a new cost function

Example

Let $\{y_t\}_t$ denote a 1D piecewise stationary random process. Assume that the y_t are independent and exponentially distributed with a scale parameter that shifts at some unknown instants $t_1, t_2, \dots$ The change points estimates are the minimizers of the negative log-likelihood, and the associated cost function is given by

$$c(y_I) = |I| \log \bar{\mu}_I$$

where I, y_I and $\bar{\mu}_I$ are respectively an interval, the sub-signal on this interval and the empirical mean of this sub-signal. The following code implements this cost function:

```
from math import log
from ruptures.base import BaseCost

class MyCost(BaseCost):

    """Custom cost for exponential signals."""

    # The 2 following attributes must be specified for compatibility.
    model = ""
    min_size = 2
```

```
    def fit(self, signal):
        """Set the internal parameter."""
        self.signal = signal
        return self

    def error(self, start, end):
        """Return the approximation cost on the segment [start:end].

        Args:
            start (int): start of the segment
            end (int): end of the segment

        Returns:
            float: segment cost
        """
        sub = self.signal[start:end]
        return (end-start)*log(sub.mean())
```

This cost function can now be used with all algorithms from `ruptures`. For instance,

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
a = np.random.exponential(scale=1, size=100)
b = np.random.exponential(scale=2, size=200)
signal, bkps = np.r_[a, b, a], [100, 300, 400]
# cost
algo = rpt.Pelt(custom_cost=MyCost()).fit(signal)
my_bkps = algo.predict(pen=10)
# display
rpt.display(signal, bkps, my_bkps)
plt.show()
```

2.4 Synthetic signals

ruptures.datasets is designed to simplify synthetic signal generation.

2.4.1 Mean shift

Description

For a given number of samples T, number of changepoints K and noise variance σ^2, this function generates change point indexes $0 < t_1 < \cdots < t_K < T$ and a piecewise constant signal $\{y_t\}_t$ with additive Gaussian noise.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
```

```
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
rpt.display(signal, bkps)
```

The mean shift amplitude is uniformly drawn from an interval that can be changed through the keyword `'delta'`.

```
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma, delta=(1, 10))
```

Code explanation

ruptures.datasets.pw_constant.**pw_constant**(*n_samples=200*, *n_features=1*, *n_bkps=3*, *noise_std=None*, *delta=(1, 10)*)

Return a piecewise constant signal and the associated changepoints.

Parameters

- **n_samples** (*int*) – signal length
- **n_features** (*int, optional*) – number of dimensions
- **n_bkps** (*int, optional*) – number of changepoints
- **noise_std** (*float, optional*) – noise std. If None, no noise is added
- **delta** (*tuple, optional*) – (delta_min, delta_max) max and min jump values

Returns signal of shape (n_samples, n_features), list of breakpoints

Return type tuple

2.4.2 Shift in correlation

Description

This function simulates a 2D signal of Gaussian i.i.d. random variables with zero mean and covariance matrix alternating between $[[1, 0.9], [0.9, 1]]$ and $[[1, -0.9], [-0.9, 1]]$ at every change point.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n = 500, 3  # number of samples
n_bkps = 3  # number of change points, noise standart deviation
signal, bkps = rpt.pw_normal(n, n_bkps)
rpt.display(signal, bkps)
```

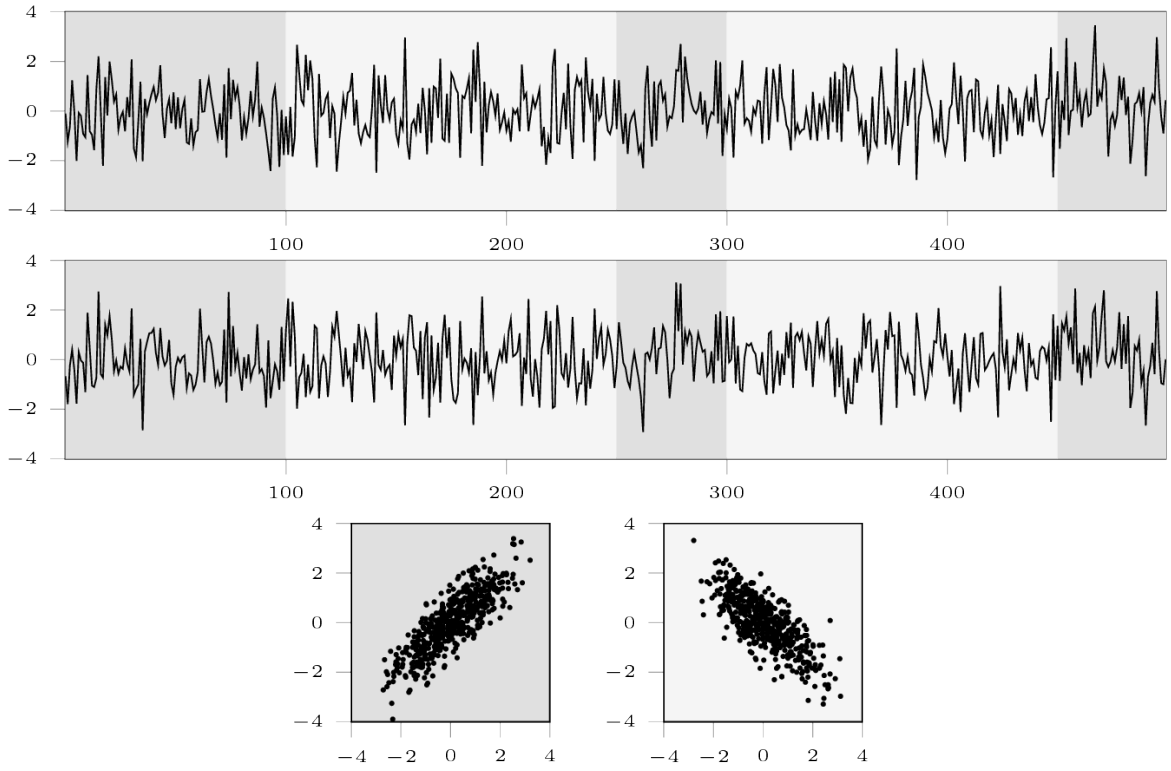


Fig. 2.4: Top and middle: 2D signal example. Bottom: Scatter plot for each regime type.

Code explanation

ruptures.datasets.pw_normal.**pw_normal**(*n_samples=200*, *n_bkps=3*)

Return a 2D piecewise Gaussian signal and the associated changepoints.

Parameters

- **n_samples** (*int, optional*) – signal length
- **n_bkps** (*int, optional*) – number of change points

Returns signal of shape (n_samples, 2), list of breakpoints

Return type tuple

2.4.3 Shift in linear model

Description

This function simulates a piecewise linear model (see *Linear model change*). The covariates standard Gaussian random variables. The response variable is a (piecewise) linear combination of the covariates.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension of the covariates
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_linear(n, dim, n_bkps, noise_std=sigma)
rpt.display(signal, bkps)
```

Code explanation

ruptures.datasets.pw_linear.**pw_linear**(*n_samples=200*, *n_features=1*, *n_bkps=3*, *noise_std=None*)

Return piecewise linear signal and the associated changepoints.

Parameters

- **n_samples** (*int, optional*) – signal length
- **n_features** (*int, optional*) – number of covariates
- **n_bkps** (*int, optional*) – number of change points
- **noise_std** (*float, optional*) – noise std. If None, no noise is added

Returns signal of shape (n_samples, n_features+1), list of breakpoints

Return type tuple

2.4.4 Shift in frequency (sine waves)

Description

This function simulates a sum-of-sine signal $y_t = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$ where $t = 0, \dots, T-1$. The frequency vector $[f_1, f_2]$ alternates between $[0.075, 0.1]$ and $[0.1, 0.125]$ at each change point index. Gaussian white noise can be added to the signal.

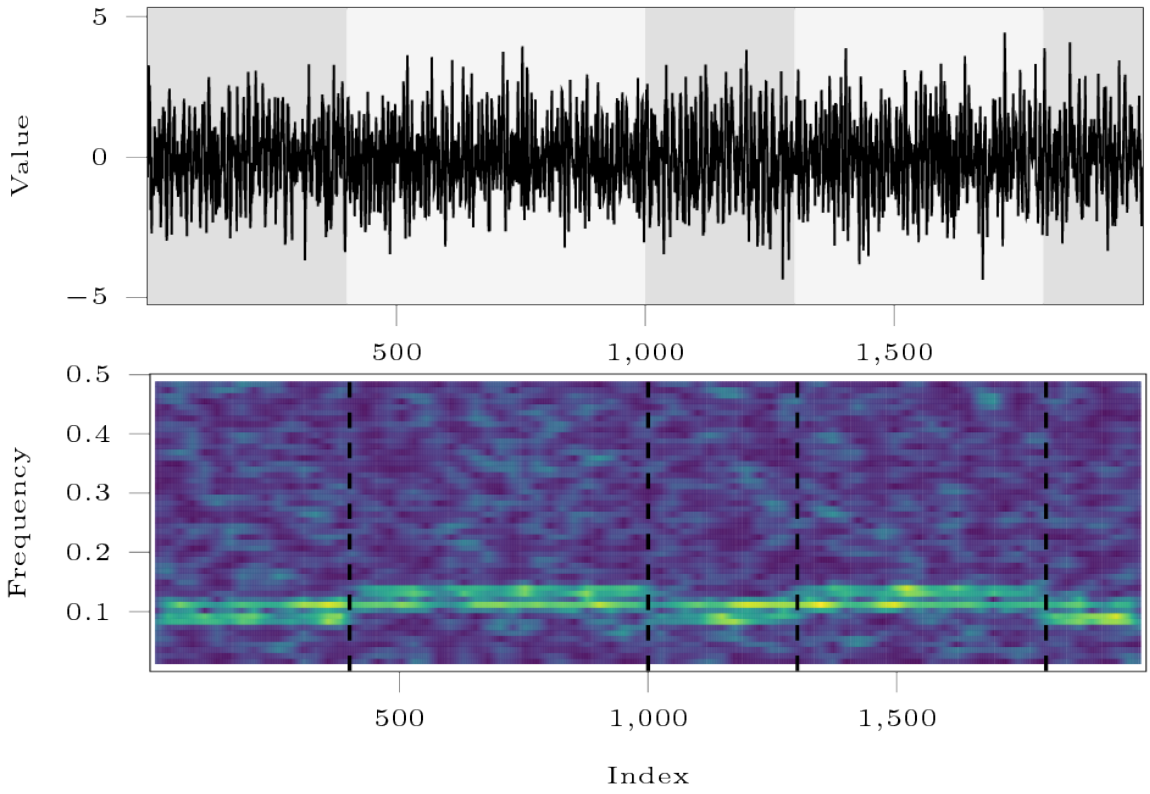


Fig. 2.5: Top: signal example. Bottom: associated spectrogram.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 3  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_wavy(n, n_bkps, noise_std=sigma)
rpt.display(signal, bkps)
```

Code explanation

ruptures.datasets.pw_wavy.**pw_wavy**(*n_samples=200*, *n_bkps=3*, *noise_std=None*)
Return a 1D piecewise wavy signal and the associated changepoints.

Parameters

- **n_samples** (*int, optional*) – signal length
- **n_bkps** (*int, optional*) – number of changepoints
- **noise_std** (*float, optional*) – noise std. If None, no noise is added

Returns signal of shape (n_samples, 1), list of breakpoints

Return type tuple

2.5 Evaluation

ruptures.metrics provides metrics to evaluate change point detection performances and ruptures.show provides a display function for visual inspection.

2.5.1 Hausdorff metric

Description

The Hausdorff metric measures the worst prediction error. Assume a set of change point indexes $t_1, t_2, \dots$ and their estimates $\hat{t}_1, \hat{t}_2, \dots$. The Hausdorff metric is then equal to

$$\text{Hausdorff}(\{t_k\}_k, \{\hat{t}_k\}_k) := \max\{\max_k \min_l |t_k - \hat{t}_l|, \max_k \min_l |\hat{t}_k - t_l|\}.$$

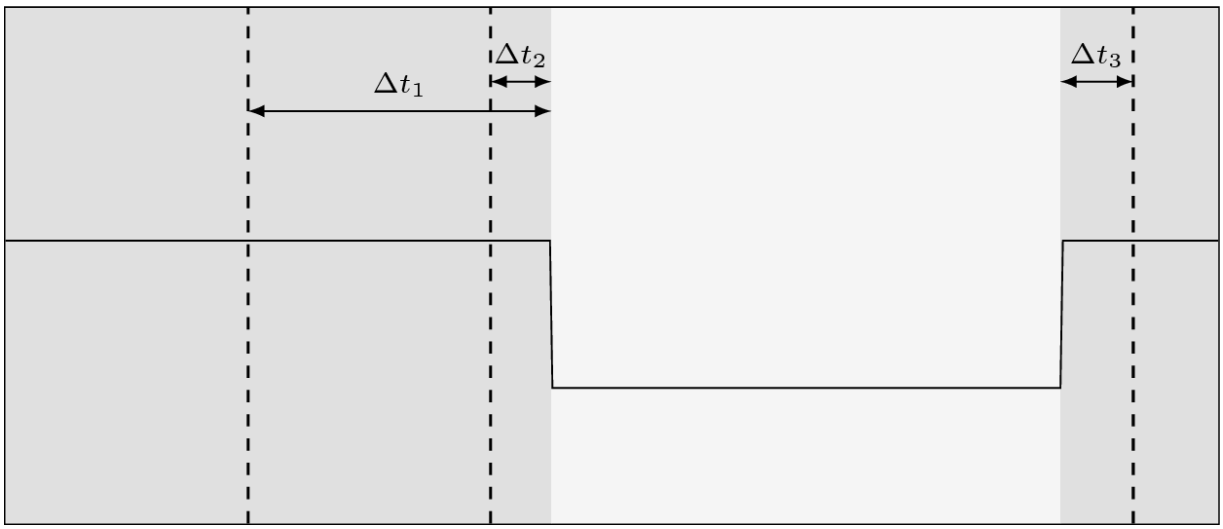


Fig. 2.6: Schematic example: true segmentation in gray, estimated segmentation in dashed lines. Here, Hausdorff is equal to $\max(\Delta t_1, \Delta t_2, \Delta t_3)$.

Usage

Start with the usual imports and create two segmentations to compare.

```
from ruptures.metrics import hausdorff
bkps1, bkps2 = [100, 200, 500], [105, 115, 350, 400, 500]
print(hausdorff(bkps1, bkps2))
```

Code explanation

ruptures.metrics.hausdorff.**hausdorff**(*bkps1*, *bkps2*)
Compute the Hausdorff distance between changepoints.

Parameters

- **bkps1** (*list*) – list of the last index of each regime.
- **bkps2** (*list*) – list of the last index of each regime.

Returns Hausdorff distance.

Return type float

2.5.2 Rand index

Description

The Rand index measures the similarity between two segmentations. Formally, for a signal $\{y_t\}_t$ and a segmentation $\mathcal{S}$, denote by A the associated membership matrix:

$$\begin{aligned} \mathcal{A}_{ij} &= 1 \text{ if both samples } y_i \text{ and } y_j \text{ are in the same segment according to } \mathcal{S} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Let $\hat{\mathcal{S}}$ be the estimated segmentation and $\hat{A}$, the associated membership matrix. Then the Rand index is equal to

$$\frac{\sum_{i<j} \mathbb{1}(A_{ij} = \hat{A}_{ij})}{T(T-1)/2}$$

where T is the number of samples. It has a value between 0 and 1: 0 indicates that the two segmentations do not agree on any pair of points and 1 indicates that the two segmentations are exactly the same.

Usage

Start with the usual imports and create two segmentations to compare.

```
from ruptures.metrics import randindex
bkps1, bkps2 = [100, 200, 500], [105, 115, 350, 400, 500]
print(randindex(bkps1, bkps2))
```

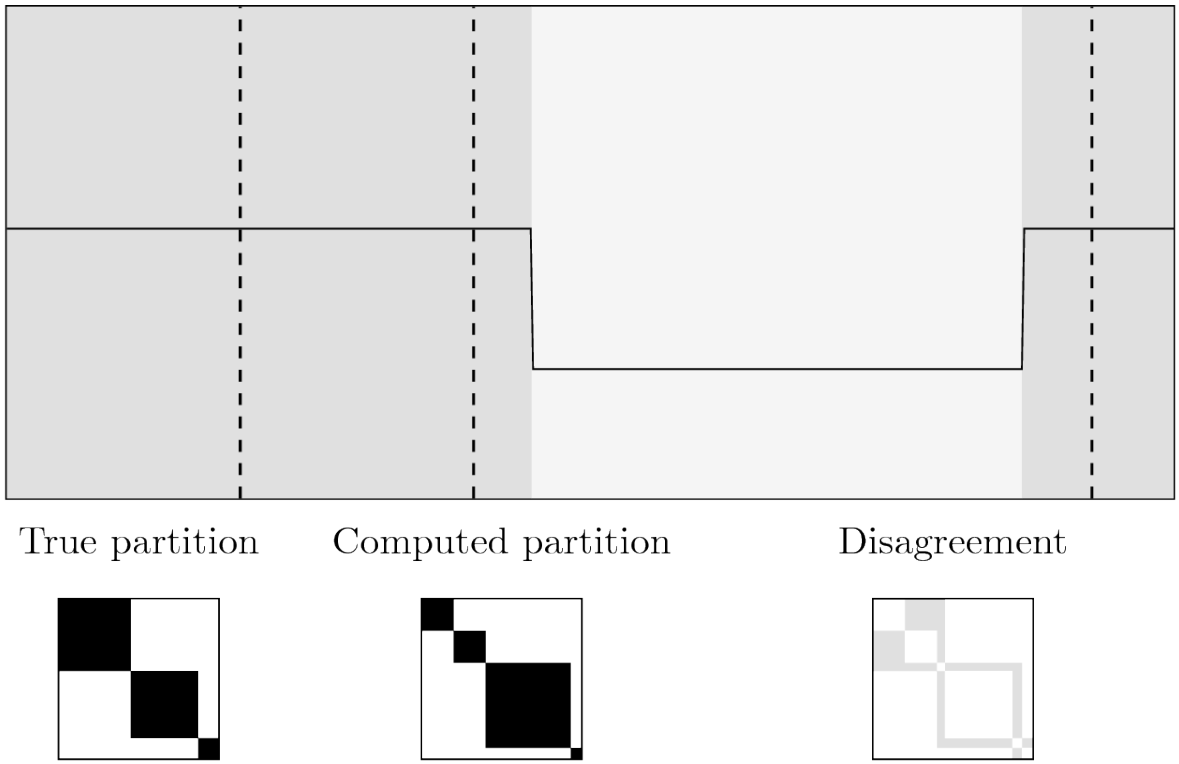


Fig. 2.7: Schematic example: true segmentation in gray, estimated segmentation in dashed lines and their associated membership matrices. Rand index is equal to 1 minus the gray area.

Code explanation

ruptures.metrics.randindex.**randindex**(*bkps1*, *bkps2*)
Rand index for two partitions. The result is scaled to be within 0 and 1.

Parameters

- **bkps1** (*list*) – list of the last index of each regime.
- **bkps2** (*list*) – list of the last index of each regime.

Returns Rand index

Return type float

2.5.3 Precision and recall

Description

A true changepoint is declared "detected" (or positive) if there is at least one computed changepoint at less than "margin" points from it. Formally, assume a set of change point indexes $t_1, t_2, \dots$ and their estimates $\hat{t}_1, \hat{t}_2, \dots$ In the context of change point detection, precision and recall are defined as follows:

$$\text{precision} := |\text{TP}|/|\{\hat{t}_l\}_l| \quad \text{and} \quad \text{recall} := |\text{TP}|/|\{t_k\}_k|$$

where, for a given margin M, true positives TP are true change points for which there is an estimated one at less than M samples, *i.e*

$$\text{TP} := \{t_k \,|\, \exists\, \hat{t}_l \text{ s.t. } |\hat{t}_l - t_k| < M\}.$$

Usage

Start with the usual imports and create two segmentations to compare.

```
from ruptures.metrics import precision_recall
bkps1, bkps2 = [100, 200, 500], [105, 115, 350, 400, 500]
p, r = precision_recall(bkps1, bkps2)
print((p, r))
```

The margin paramater M can be changed through the keyword `'margin'` (default is 10 samples).

```
p, r = precision_recall(bkps1, bkps2, margin=10)
print((p, r))
p, r = precision_recall(bkps1, bkps2, margin=20)
print((p, r))
```

Code explanation

ruptures.metrics.precisionrecall.**precision_recall**(*true_bkps*, *my_bkps*, *margin=10*)
Calculate the precision/recall of an estimated segmentation compared with the true segmentation.

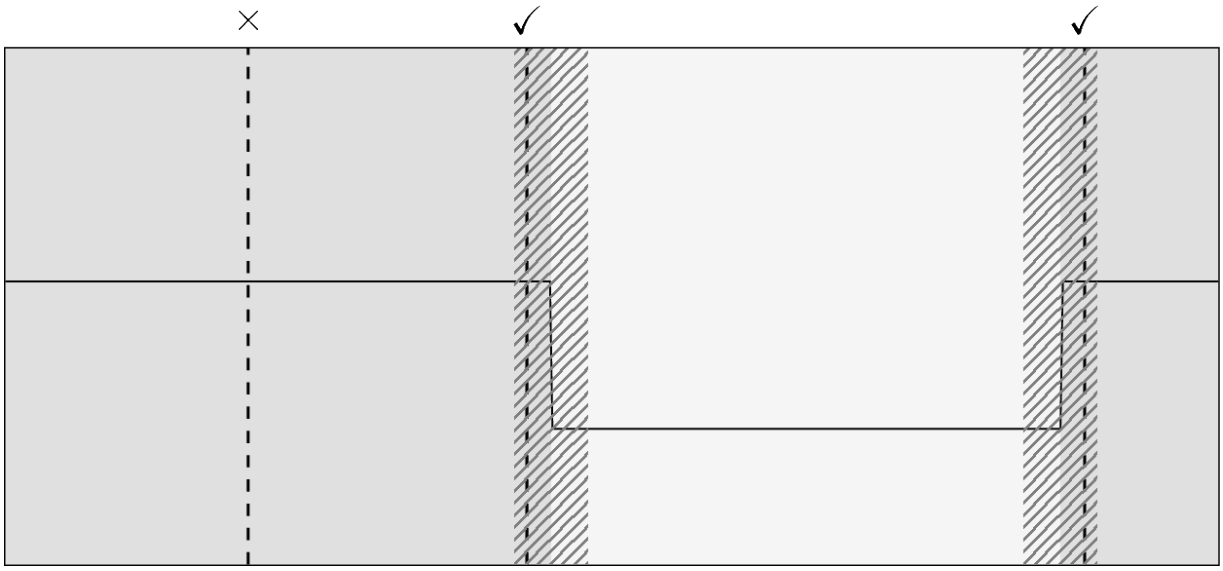


Fig. 2.8: Schematic example: true segmentation in gray, estimated segmentation in dashed lines and margin in dashed areas. Here, precision is 2/3 and recall is 2/2.

Parameters

- **true_bkps** (*list*) – list of the last index of each regime (true partition).
- **my_bkps** (*list*) – list of the last index of each regime (computed partition).
- **margin** (*int, optional*) – allowed error (in points).

Returns (precision, recall)

Return type tuple

2.5.4 Display

Description

The function *display()* displays a signal and the change points provided in alternating colors. If another set of change point indexes is provided, they are displayed with dashed vertical dashed lines.

Usage

Start with the usual imports and create a signal.

```
import numpy as np
import matplotlib.pylab as plt
import ruptures as rpt
# creation of data
n, dim = 500, 2  # number of samples, dimension
n_bkps, sigma = 3, 5  # number of change points, noise standart deviation
signal, bkps = rpt.pw_constant(n, dim, n_bkps, noise_std=sigma)
rpt.display(signal, bkps)
```

If we computed another set of change points, for instance `[110, 150, 320, 500]`, we can easily compare the two segmentations.

```
rpt.display(signal, bkps, [110, 150, 320, 500])
```

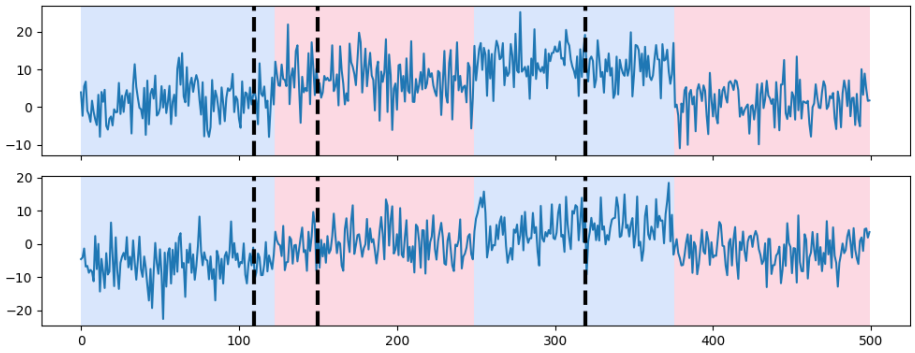


Fig. 2.9: Example output of the function *display()*.

Code explanation

ruptures.show.display.**display**(*signal*, *true_chg_pts*, *computed_chg_pts=None*, ***kwargs*)

Display a signal and the change points provided in alternating colors. If another set of change point is provided, they are displayed with dashed vertical dashed lines.

Parameters

- **signal** (*array*) – signal array, shape (n_samples,) or (n_samples, n_features).
- **true_chg_pts** (*list*) – list of change point indexes.
- **computed_chg_pts** (*list, optional*) – list of change point indexes.

Returns (figure, axarr) with a `matplotlib.figure.Figure` object and an array of Axes objects.

Return type tuple

CHAPTER

THREE

CONTACT

Charles Truong.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[CTOV18] C. Truong, L. Oudre, and N. Vayatis. A review of change point detection. *arXiv preprint arXiv:1801.00718*, pages 1–31, 2018. arXiv:1801.00718.

[BKFE12] R. Killick, P. Fearnhead, and I. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[BSBai97] J. Bai. Estimating multiple breaks one at a time. *Econometric Theory*, 13(3):315–352, 1997.

[BSFry14] P. Fryzlewicz. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014. doi:10.1214/14-AOS1245.

[BUFry07] Piotr Fryzlewicz. Unbalanced Haar Technique for Nonparametric Function Estimation. *Journal of the American Statistical Association*, 102(480):1318–1327, 2007. doi:10.1198/016214507000000860.

[BUKCHP01] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 289–296. 2001.

[C1Bai95] J. Bai. Least absolute deviation of a shift. *Econometric Theory*, 11:403–436, 1995.

[KERACH12] S. Arlot, A. Celisse, and Z. Harchaoui. Kernel change-point detection. *arXiv preprint arXiv:1202.3878*, 1(0000):1–26, 2012.

[KERGBR+12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

[CLBP03] J. Bai and P. Perron. Critical values for multiple structural change tests. *Econometrics Journal*, 6(1):72–78, 2003.

[ARBai00] J. Bai. Vector autoregressive models with structural changes in regression coefficients and in variance–covariance matrices. *Annals of Economics and Finance*, 1:303–339, 2000.

[MLXJR03] E. P. Xing, M. I. Jordan, and S. J. Russell. Distance metric learning, with application to clustering with side-Information. In *Advances in Neural Information Processing Systems 21 (NIPS 2003)*, 521–528. 2003.

PYTHON MODULE INDEX

r

INDEX

Symbols

B

C

D

E

F

H

P

R

W

B

An automated recording method in clinical consultation to rate the limp in lower limb osteoarthritis

PLOS ONE

RESEARCH ARTICLE

An Automated Recording Method in Clinical Consultation to Rate the Limp in Lower Limb Osteoarthritis

R. Barrois[1], Th. Gregory[2], L. Oudre[1,3], Th. Moreau[1], Ch. Truong[1], A. Aram Pulini[1], A. Vienne[1], Ch. Labourdette[1,4], N. Vayatis[1,4], S. Buffat[1,5], A. Yelnik[1,6], C. de Waele[1], S. Laporte[7], P. P. Vidal[1], D. Ricard[1,8]*

1 Cognition and Action Group, Cognac-G, CNRS, Université Paris Descartes, SSA, Paris, France, **2** Service de chirurgie orthopédique et traumatologie, HEGP, université Paris Descartes, Paris, France, **3** Institut Galilée, Université Paris 13, Villetaneuse, France, **4** Centre des Mathématiques et de Leurs Applications, Ecole Normale Supérieure de Cachan, Cachan, France, **5** Institut de Recherche Biomédicale des Armées, Brétigny-sur-Orge, France, **6** PRM Department, GH St Louis Lariboisière F. Widal, AP-HP, Diderot University, Paris, France, **7** LBM/Institut de Biomécanique Humaine Georges Charpak, Arts et Métiers Paris Tech, 151 Boulevard de l'Hôpital, 75003, Paris, France, **8** Service de Neurologie, Hôpital d'Instruction des Armées de Percy, Service de Santé des Armées, Clamart, France

* damien.ricard@m4x.org

Citation: Barrois R, Gregory T, Oudre L, Moreau T, Truong C, Aram Pulini A, et al. (2016) An Automated Recording Method in Clinical Consultation to Rate the Limp in Lower Limb Osteoarthritis. PLoS ONE 11(10): e0164975. doi:10.1371/journal.pone.0164975

Editor: Steven Allen Gard, Northwestern University, UNITED STATES

Received: October 7, 2015

Accepted: October 4, 2016

Published: October 24, 2016

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: SATT Innov Ile de France. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

Abstract

For diagnosis and follow up, it is important to be able to quantify limp in an objective, and precise way adapted to daily clinical consultation. The purpose of this exploratory study was to determine if an inertial sensor-based method could provide simple features that correlate with the severity of lower limb osteoarthritis evaluated by the WOMAC index without the use of step detection in the signal processing. Forty-eight patients with lower limb osteoarthritis formed two severity groups separated by the median of the WOMAC index (G1, G2). Twelve asymptomatic age-matched control subjects formed the control group (G0). Subjects were asked to walk straight 10 meters forward and 10 meters back at self-selected walking speeds with inertial measurement units (IMU) (3-D accelerometers, 3-D gyroscopes and 3-D magnetometers) attached on the head, the lower back (L3-L4) and both feet. Sixty parameters corresponding to the mean and the root mean square (RMS) of the recorded signals on the various sensors (head, lower back and feet), in the various axes, in the various frames were computed. Parameters were defined as discriminating when they showed statistical differences between the three groups. In total, four parameters were found discriminating: mean and RMS of the norm of the acceleration in the horizontal plane for contralateral and ipsilateral foot in the doctor's office frame. No discriminating parameter was found on the head or the lower back. No discriminating parameter was found in the sensor linked frames. This study showed that two IMUs placed on both feet and a step detection free signal processing method could be an objective and quantitative complement to the clinical examination of the physician in everyday practice. Our method provides new automatically computed parameters that could be used for the comprehension of lower limb osteoarthritis. It may not only be used in medical consultation to score patients but also to monitor the evolution of their clinical syndrome during and after rehabilitation. Finally, it

paves the way for the quantification of gait in other fields such as neurology and for monitoring the gait at a patient's home.

Introduction

Gait analysis plays an important role in the study of lower limb osteoarthritis on two grounds: first, osteoarthritis has important repercussions on gait biomechanics [1–4]. It rapidly worsen the prognosis for the affected joints, and on the long term affect the intact ones, which further compromises the mobility of the patients. Second, the functional syndrome, *ie* the limp evaluated with infrared markers, is well correlated with the severity of the pathology [5]. By using stereophotogrammetry and force plates in gait laboratories, compared to matched controls, knee osteoarthritis patients had reductions in walking speed [6–8], lower cadence [9,10], longer double support time [9,11] and a smaller stride length [12]. That is, gait analysis would be useful to quantify precisely the severity of osteoarthritis in a given patient. However, until recently, gait laboratories were too expensive and complex to be utilized in daily practice. This explains that clinical scores remains the gold standard to evaluate the severity of the pathology up to these days [13–15]. The Western Ontario and MACmaster Universities osteoarthritis index (WOMAC) is actually the most largely used of these scores in rheumatology for lower limb osteoarthritis to assess pain, stiffness, and physical function in patients. WOMAC is considered to be reliable, sensitive and adapted to clinical practice [16–18] and therefore, it is used in most osteoarthritis clinical studies [19,20]. It remains that clinical scores are inherently subjective, as they are based on the patient's verbal reports and on the clinician's visual skills and interpretations. For instance, the WOMAC index does not accurately reflect walking performances [21,22] and clinical scores have a lack of sensitivity for identifying changes of balance and walking in mild to moderate disease severity [23].

In that context, skin-mounted accelerometers seem to be well-suited for investigating gait kinematics in osteoarthritis patients [24]. They are inexpensive and non-invasive devices and, more importantly, they are suited for routine clinical practice. In particular, they can be used to evaluate gait using a standard protocol, which involves walking ten meters forward and ten meters back on a level surface at a self-selected walking speed [25–31]. An essential point using gait analysis in the everyday consultation is to extract from the raw data, automatically and in real time, useful parameters for the clinician. To begin, step detection and gait cycle identification are critical for computing gait parameters. By hand, it is time consuming and unfit for clinical practice [3,13,32,33]. On the other hand, the automated routines available for step detection are not robust because they are based on *a priori* predetermined threshold values [34,35]. In addition, step detection automated routines are based on the assumption that steps have stable kinematics, which is not the case in pathological conditions [34,36–40].

Inertial sensors are suitable for quantifying gait performance directly at the routine consultation level. For this use, the quantification can be driven by real-time and low-powered software. Advanced trunk accelerometric parameters have been found useful for detecting pathological gait [41]. Nevertheless, complex gait parameters often require previous step detection, which requires extensive and time-consuming computation for sufficient robustness. As well, the clinical meaning of complex gait parameters is not always clear, although recent papers have made substantial efforts to clarify this point [42]. Still, this situation is unfortunate because straight-forward gait parameters (mean or root mean square [RMS]) for the signals often reveal clinically interpretable results [41,43]. Therefore, simple parameters such as the

RMS remain commonly used but often only for the lower back sensor [28,41,44–48]. They often show differences between the pathological and healthy gait. These simple parameters have not been explored at other key anatomical landmarks of the body.

Finally, the gait parameters have widely been developed in complex neurological limping models such as Parkinson disease, cerebral palsy or peripheral neuropathy and not in osteoarthritis, in which pain is believed to be the major limping cause and for which the simple gait parameters could have a direct, understandable clinical meaning [41].

Hence, we have tried to revisit the problem of gait analysis in osteoarthritis patients in daily practice using four inertial motion units (IMU) strapped to the head, lower back (L3-L4) and feet. We have also designed a new automated and online method of gait analysis. This method was then evaluated by comparing its outcome to the severity of the lower limb osteoarthritis evaluated with the WOMAC index in a cohort of 48 patients and 12 control subjects.

Methods

Subjects

All subjects (patients and control subjects) were coming for a clinical consultation at the orthopedic surgeon's office (ThG) during three consecutive months. All consecutive patients or control subjects reaching the inclusion criteria during the inclusion period were included in the study.

All patients had hip or knee osteoarthritis diagnosed by an orthopedic surgeon (ThG) and graded with the WOMAC index (0 to 96). Patients had neither vestibular, neurological, or musculoskeletal disorders, nor any fractures of the lower extremity, nor rheumatoid arthritis or generalized osteoarthritis. Forty-eight patients with lower limb osteoarthritis were included (43 to 90 years, mean 70.9 years). Patients were divided into 2 severity groups of equal size separated by the median of the WOMAC index: the moderately impaired group (G 1) and the severely impaired group (G 2). The median value of the WOMAC index was 45/96. This median-based repartition was chosen in order to maximize the power of the statistical analysis.

The control subjects had no orthopedic nor neurological problem that could affect their gait pattern. Twelve control subjects were included (40 to 87 years, mean 60.8). They formed the age-matched control group (G 0). The mean and standard deviation (SD) of the age, body mass index (BMI) and WOMAC index of each group are shown in Table 1.

To assess the test–retest validity of the discriminating parameters, we checked their variability with IMU placement. For the sensor-placement control experiment 1, 2 healthy controls (age 22 and 23 years) performed 5 walking trials with sensors placed by 2 different operators at each trial. For the sensor-placement control experiment 2, these 2 subjects also performed 9 walking trials with displacement of the sensor along the antero-posterior (AP) axis and the

Table 1. Age body mass index (BMI) and WOMAC index mean (upper case) and standard deviation (lower case) of group 1 and group 2 patients with symptomatic lower limb osteoarthritis and age matched controls.

Group	Number	Age	BMI	WOMAC
0	12	63,2	25,2	0,0
		17,1	4,6	0,0
1	24	70.5	26.8	14,1
		9.5	5,7	10,0
2	24	70,5	28,2	62,58
		14,9	5,5	14,0

doi:10.1371/journal.pone.0164975.t001

medio-lateral (ML) axis in terms of the reference position (from -20 to +20 mm in 5-mm increments). Coefficients of variation ($CV = \frac{\mu}{\sigma}$) were evaluated for these 2 experiments, where μ is the mean and σ the standard deviation of the parameters over all trials for each sensor control experiment. A CV $< 5\%$ was considered correct and $< 10\%$ acceptable.

The study was validated by a local ethic comity (Comité de Protection des Personnes Ile de France II, n°CPP 2014-10-04 RNI) and both patients and control subjects gave their written consent to participate.

Instrumentation

Linear accelerations and angular velocities of the head, lower back (L4-L5 vertebra) and feet were collected using four IMUs including triaxial accelerometers, gyroscopes and magnetometers (XSens®, Culver City, CA, USA, MTw Measurement Units, 3,5h LiPo battery, 27g, 3,5x5,8x1,0cm^3, +/-16g, +/-1200deg/s, 100Hz, errors 0,003m/s^2 and 0,05deg/s), fixed with manufacturer-designed adhesive straps and connected through WiFi with a computer.

Defining the sensor linked frame and the doctor's office linked frame

The accelerations and the angular velocities of the four IMUs can be expressed in the sensor linked frame and in the doctor's office linked frame.

The IMUs were fixed and aligned with respect to the body in the following way. The head sensor was positioned on the center of the forehead. The antero-posterior (AP) axis of the frame linked to the head sensor was the normal to the forehead surface. The medio-lateral (ML) axis was set parallel to the line joining the left temple and the right temple. The vertical (V) axis completed the orthonormal frame. The lumbar sensor was positioned at L4-L5 level. The AP axis of the frame linked to the lumbar sensor was normal to the back surface. The ML axis was set parallel to the line joining the right anterior superior iliac spine and left anterior superior iliac spine. The V axis completed the orthonormal frame. Each foot sensor was positioned at the center of the dorsal face of each foot. The V axes of each frame linked to each foot sensor were the normals to the dorsal surfaces of each foot. The AP axis was set parallel to the longitudinal direction of the foot. The ML axis completed the orthonormal frame. Positive directions for the axes were not defined because all computed gait parameters are independent of this orientation.

The doctor's office frame was the fix frame linked to the doctor's office. The V axis of the doctor's office linked frame was aligned with the gravity. The horizontal plane (H) was the plane normal to the V axis. AP and ML axes were not defined in the doctor's office linked frame. The change of frame from the sensor linked frames to the doctor's office linked frame was done with an algorithm [49,50] based on the XSens® 3D magnetometer measurement. We used the manufacturer's rotation matrix as described and validated by Cognolato [50].

Experimental design and data acquisition

The WOMAC index was evaluated and recorded by the same experimented orthopedic surgeon (ThG). The questions were always asked in the same order with the validated text. After the sensor fixation, the participant was instructed to execute the following steps: stand quiet for six seconds, walk ten meters at a preferred walking speed, make a U-turn, walk the ten meters back and stand quiet for two seconds.

Participants could keep their clothes and their shoes on. Participants with high heels (>2 cm) were asked to do the exercise without their shoes. Each participant completed two trials of this exercise to improve the reliability of the measure.

Data processing

Each phase of the exercise (quiet standing, walking and U-turn) was manually annotated without any step detection (RB). All parameters were computed on the concatenated signal of the walk phases of the exercise (Fig 1). One given parameter p is defined by a *sensor* = {*head*, *lower back*, *ipsilateral foot*, *contralateral foot*}, a *frame* = {*sensor*, *office*}, an *axis* = {*AP*,*ML*,*V*} if the frame is the sensor-linked frame or an *axis* = {*H*, *V*} if the frame is the doctor's office-linked frame (H for horizontal plane), a signal *sig* = {*acceleration*, *angular velocity*} and a statistical tool *stat* = {*mean*, *RMS*}. Thus we computed the following:

$$p_{sensor,frame,axis,sig,mean} = mean|sig_{sensor,frame,axis}|$$

$$p_{sensor,frame,axis,sig,RMS} = RMS|sig_{sensor,frame,axis}|$$

where |.| is the absolute value and where in the case of n values $x = \{x_1,x_2,\ldots,x_n\}$:

- the mean is defined by $mean(x) = \frac{1}{n}(x_1 + x_2 + \cdots + x_n)$
- the RMS is defined by $RMS(x) = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \cdots + x_n^2)}$

For each parameter, the mean of the two trials was taken. Sixty parameters were computed, fifteen for each sensor (Table 2).

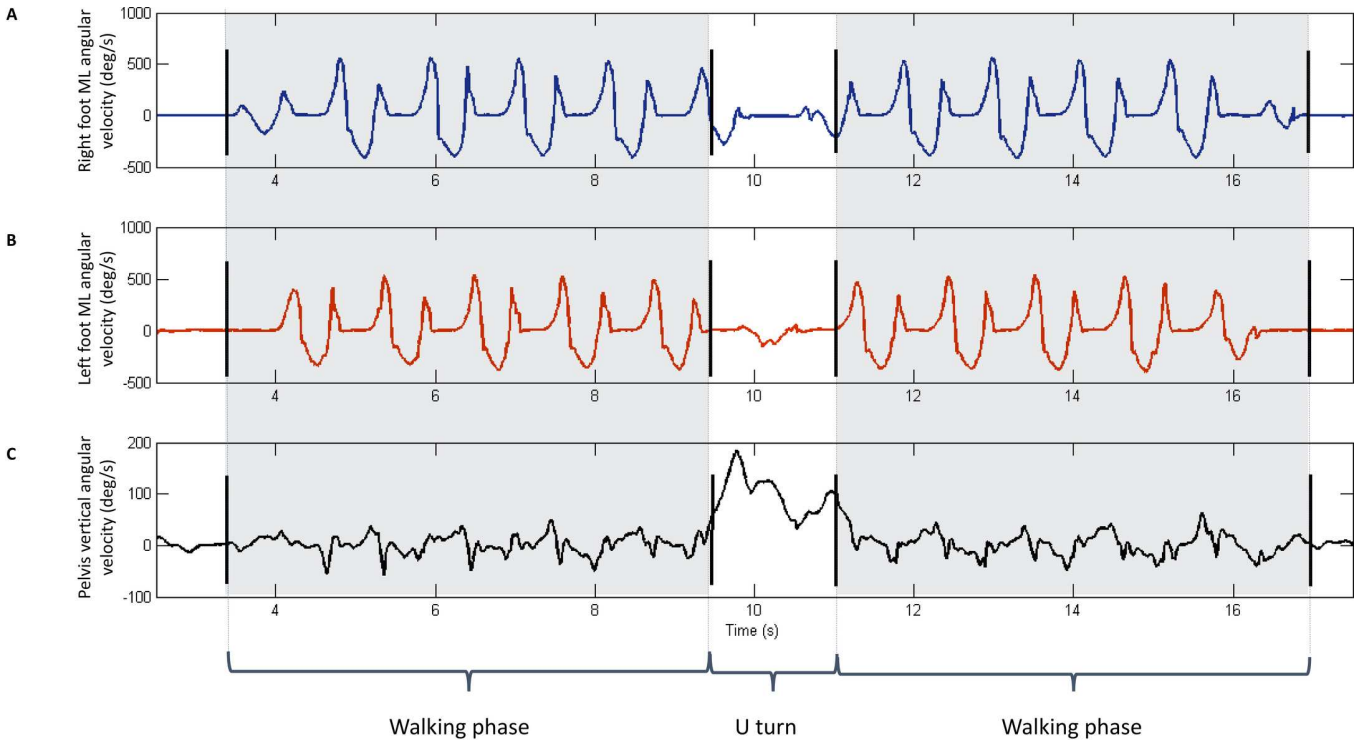


Fig 1. Representative data and manual phase annotation result for one healthy participant performing a 10 meters go and 10 meters back walking exercise at self-selected walking speed. Black bars stand for manual annotation. Dashed zone corresponds to the walking phases. The walking parts of the signal were taken for parameter computation. (**A**)–Representative ML lateral angular velocity in the sensor linked frame for right foot. (**B**)—Representative ML lateral angular velocity in the sensor linked frame for left foot. (**C**)–Representative V angular velocity in the sensor linked frame for L3-L4.

doi:10.1371/journal.pone.0164975.g001

Table 2. Acceleration and angular velocity parameters in the sensor linked frames and the doctor's office linked frame. RMS for root mean square.

Sensor linked frame		
Axis and plane	**Acceleration**	**Angular velocity**
Meadial lateral (ML)	-	Mean
	RMS	RMS
Anterior posterior (AP)	-	Mean
	RMS	RMS
Vertical (V)	-	Mean
	RMS	RMS
Doctor's office linked frame		
Axis and plane	**Acceleration**	**Angular velocity**
Horizontal (H) plane	Mean	-
	RMS	-
Vertical (V) axis	Mean	Mean
	RMS	RMS

doi:10.1371/journal.pone.0164975.t002

The parameters were also computed by sliding the manually-annotated computation window one second earlier and one second later to take the error of the manual phase annotation into consideration (see Results section). The parameters affected by the gravity component were not studied because they were too sensor's positioning dependent. These parameters were: the mean of the norm of the acceleration in the sensor linked frames in the AP, ML and V directions on the four markers *i.e.*:

$$p_{\{head,lower\ \ back,feet\},sensor,\{AP,ML,V\},acceleration,mean}$$

Gravity component of the acceleration was not removed. The angular velocities in the horizontal plane in the doctor's office frame was not studied because of the absence of clinical meaning of this parameter *i.e.*:

$$p_{\{head,lower\ \ back,feet\},office,H,angular\ \ velocity,\{mean,RMS\}}$$

Mean walking velocity was computed by dividing the walking distance (20 m) by the duration of the walking phases.

Statistical analysis

A one-way analysis of variance (ANOVA) with Tukey pairwise comparison test and a one-way analysis of covariance (ANCOVA) with age and BMI as covariate with Tukey pairwise comparison were performed on all three groups on all the 61 parameters. Mean walking velocity was not taken as covariate because it is known to decrease with lower limb osteoarthritis severity [25]. We defined a discriminating parameter as a parameter that showed statistical differences using an ANOVA analysis with a Tukey pairwise comparison test (p-value set under 0, 05) between all three groups (G1vsG2, G2vsG3 and G1vsG3).

Results

Data processing

We could manually annotate the initial quiet-standing phase, the go-walking phase, the U-turn and the back-walking phase for all 48 lower limb osteoarthritis patients and the 12 control subjects. Representative data and manual phase annotation results for one control subject

performing a 10 meters forward and 10 meters back walking task at a self-selected walking speed are shown in Fig 1. The cumulative error for the manual exercise phase annotation was 1 second. The relative errors due to the manual annotation error on the parameters were 5% on average for the mean of the acceleration in the horizontal plane on the ipsilateral foot. The errors did not change the statistical significance of the in-between group differences shown by the discriminating parameters.

Parameters and statistical analysis

Looking at the 60 IMU-based parameters we found (S1 Table):

- in the sensor linked frames: no discriminating parameters (results not shown).
- in the doctor's office linked frame: the mean and the RMS of the norm of the acceleration in the horizontal plane for the contralateral (p-values respectively G0vsG1 = 0.011; G1vsG2 = 0.013; G0vsG2<0.0001 for mean and G0vsG1 = 0.010 G1vsG2 = 0.026; G0vsG2<0.0001 for RMS) and the ipsilateral (p-values respectively G0vsG1 = 0.002; G1vsG2 = 0.0004; G0vsG2<0.0001 for mean and; G0vsG1 = 0.001; G1vsG2 = 0.001; G2vsG0<0.0001 for RMS) foot were discriminating parameters (Fig 2). In our predefined formalism these parameters are $p_{\{ipsilateral\ foot,controlateral\ foot\},office,H,acceleration,\{mean,RMS\}}$.

- These parameters can be

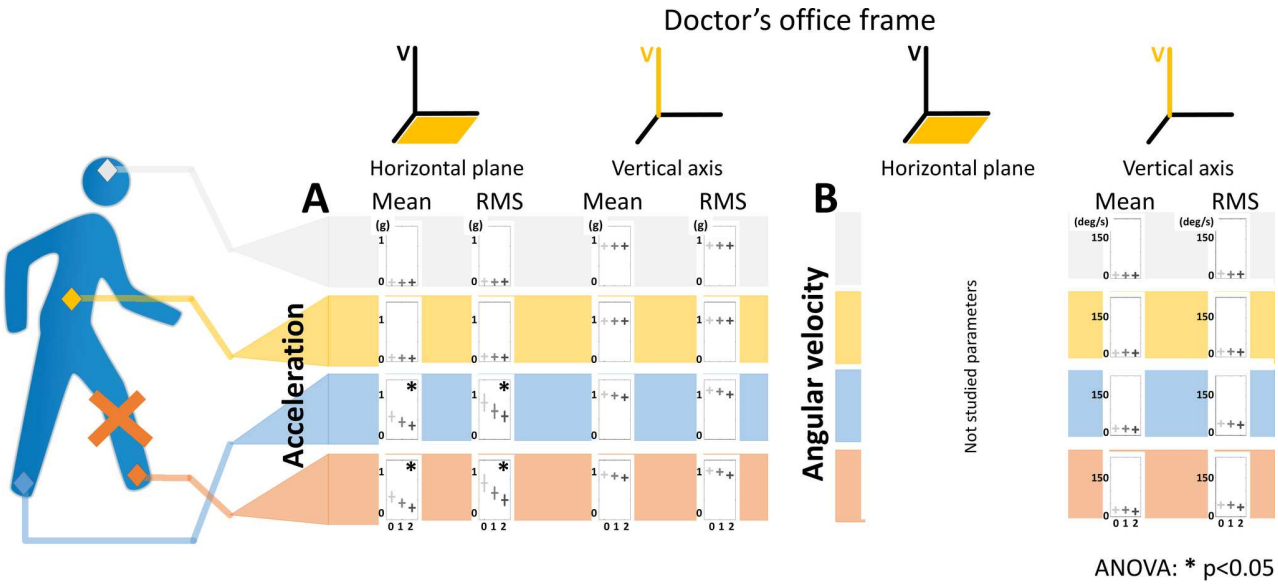


Fig 2. Selected 24 parameters out of the 60 IMU based parameters computed in the doctor's office linked frame obtained from 4 IMUs on 12 control subjects and 48 patients during a 10 meters go and 10 meters back walking task. Sensor location are shown on the walking silhouette by colored diamonds: grey for the head, yellow for the sacrum, blue for the contralateral foot and red for the ipsilateral foot. The red cross of the walking silhouette indicates the ipsilateral foot to the lesion defined by the side where the patient is the more symptomatic. Each parameter is represented by a bar diagram. The row indicate the location of the sensor and whether the parameters is computed on an acceleration (**A**) or an angular velocity signal (**B**). The columns indicate whether the parameter is computed on the horizontal plane or on the vertical axis and whether the parameter is a mean or a RMS of the norm of the walking signal. In each bar diagram, the parameter is represented as a function of the severity. The results are shown by a modulated grey cross: horizontal bar stands for mean and vertical bar stands for the standard deviation. Light grey represents the healthy group (G0), medium grey the moderately impaired group (G1) and dark grey the severely impaired group (G2). The parameters marked by a star (*) are the discriminating parameters (parameters that show significant difference between the three WOMAC index defined severity groups). The statistical analysis was performed with an ANOVA analysis and a Tukey pairwise comparison test (p-value set at 0.05). RMS stands for root mean square and V for vertical axis.

doi:10.1371/journal.pone.0164975.g002

In the sensor linked frame, angular velocities around the ML axis on the ipsilateral and contralateral feet didn't appear to be discriminating parameters, but showed statistical significant differences between the group of control subjects and the two groups of patients (results not shown).

No parameters from the lower back and no parameter from the head were discriminating parameters.

We found that the mean and RMS of the norm of the acceleration in the horizontal plane in the doctor's office linked frame for contralateral and ipsilateral feet still met our definition of discriminating parameters with age and BMI as covariate.

For walking velocity, differences were significant between G0 and G2, G1 and G2. No significant difference in walking velocity was found between G0 and G1 (Fig 3). Thus, walking velocity was not a discriminating parameter.

Sensor-placement control experiment 1 gave a CV $< 5\%$ and experiment 2 a CV $< 10\%$ for the mean of the norm of the acceleration in the horizontal plane and the RMS of the norm of the acceleration in the horizontal plane (Table 3).

Discussion

The correlation between lower limb osteoarthritis severity and stereophotogrammetry is well established [5,26,27,51–57]. In contrast, only two studies retrieved the same correlation using inertial sensors [52,54]. We confirm that result here. In addition, to the best of our knowledge, it is the first lower limb osteoarthritis study where the IMU-based gait parameters were extracted without step detection, which is important for daily clinical use. Finally, our results suggest that two IMUs placed on the feet are sufficient to quantify the severity of inferior limb osteoarthritis, which further improves the use of the method in daily practice.

We compared 48 patients and 12 control subjects walking 10 meters forward and 10 meters back under clinical consultation conditions. The four-IMUs-based method showed a discrimination capacity of clinical severity groups for 4 of the 60 parameters tested. These discriminating parameters were: mean and RMS of the norm of the acceleration in the horizontal plane in the doctor's office linked frame for the contralateral and the ipsilateral feet. The results remained statistically significant with BMI and age as covariate. The absence of clinical correlation with parameters in the head and lower back reflected that lower limb osteoarthritis impacted the kinematics of the painful segment more than the upper body, which, to the best of our knowledge, has not been specifically shown previously [1,12,56,58–61]. However, it cannot be excluded that a more precise method of measurement, such as stereophotogrammetry, could reveal subtle differences. It remains that one important conclusion would be that two sensors placed at the feet, would be sufficient in daily practice to rate osteoarthritis severity.

Walking speed is known to influence gait parameters [62] and osteoarthritis reduces walking speed. Hence, the question is whether the influence of osteoarthritis severity on the gait parameters was solely caused by the reduction of walking speed, or if osteoarthritis *per se* led to a change of gait pattern. To analyze the change of walking pattern independently from the walking speed, a first method is to walk at a predetermined walking speed [1,3,12,33,56,58,63–65]. It requires dedicated material (treadmill), which is not suited in daily clinical practice and it does not allow to capture natural and repeatable walking patterns [25]. A second method is to select subgroups of participants walking at their preferred walking speed matched in walking speed [66]. But, the subgroups do not reflect the general populations of the whole severity groups [25]. A third method would be to set walking speed as covariate [63,67,68]. As walking speed is inherently linked to disease progress, and its mean value tends to decrease with increasing levels of disease severity, this technique is inappropriate [25]. Therefore, we chose to

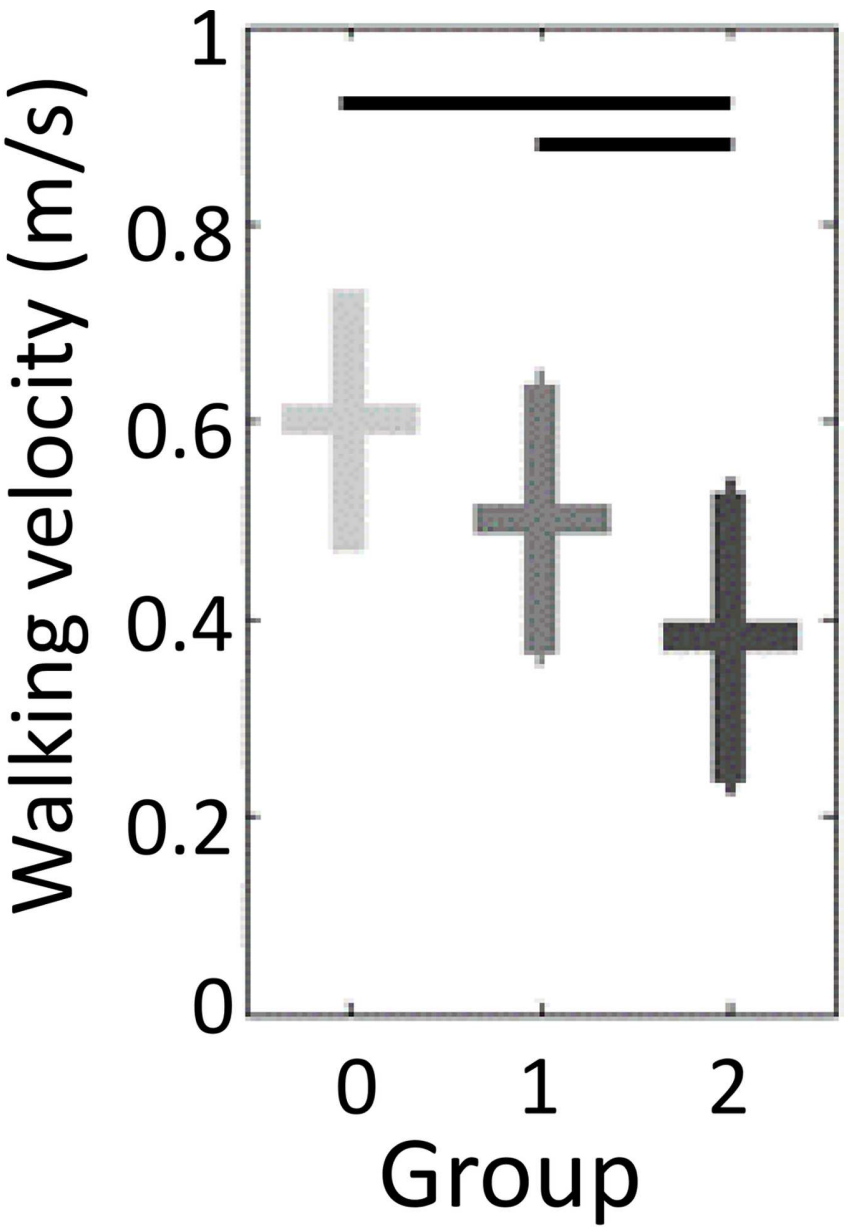


Fig 3. Mean walking velocity as a function of the WOMAC index based osteoarthritis severity groups. The results are shown by a modulated grey cross: horizontal bar stands for mean and vertical bar stands for the standard deviation. Light grey represents the healthy participants, medium grey the moderately impaired group and dark grey the severely impaired group. Black horizontal bars show the statistical differences between the groups computed with an ANOVA analysis and a Tukey pairwise comparison test (p-value set at 0.05).

doi:10.1371/journal.pone.0164975.g003

have participant walking at preferred walking speeds. Using that method, we showed on our dataset that walking velocity was not a discriminating parameter when comparing G0 and G1. Altogether, this negative result suggests that osteoarthritis *per se* caused a change of gait pattern, independent from the walking velocity. Pain could likely be a factor.

Table 3. Sensor-placement control experiment 1 (Exp. 1): coefficient of variation (CV; mean/SD) of the mean and root mean square (RMS) of the norm for acceleration in the horizontal plane in the right foot for 2 subjects over 5 walking trials with renewal of the sensor placement at each trial. Sensor-placement control experiment 2 (Exp. 2): CV over 9 walking trials (-20; -15; -10; -5; 0; 5; 10; 15; 20 mm) with displacement of the sensor in increments of 5 mm along the antero-posterior axis and medio-lateral axis in terms of the reference position. Values are in percentages.

		Mean	RMS
Exp. 1	**Subject 1**	3.5*	4.3*
	Subject 2	0.9*	2.1*
Exp. 2	**Subject 1**	7.4	8.9
	Subject 2	2.9*	4.0*

* CV < 5%.

doi:10.1371/journal.pone.0164975.t003

Our method gave a global view of the gait kinematics, which summed up the impacts of ostheoarthritis at the hip, knee and ankle joints levels. Also, ipsilateral and contralateral sides were defined with respect to the more symptomatic side of the patient. Therefore, our approach may help to objectively rate lower limb osteoarthritis severity in daily clinical practice but it is not suited to gain a detailed insight in the walking pattern of these patients [52].

The manual phase annotation of the walking exercise we used saved time but could have lowered the robustness of our method. However, we showed that the errors due to manual annotation didn't change the statistical validity of the discriminating parameters in our study. Computation of gait parameters in the sensor-linked frame is prone to lower the reproducibility of the parameters because it is biased by the inherent variability of the positions of the sensors [44,69]. This explains why in our study, robust discriminating accelerometric parameters for lower limb osteoarthritis severity were all found in the doctor's office linked frame.

Two aspects of the positioning of the sensors may affect gait parameters by using IMUs: the orientation and position of the sensor on the measured body segment [70–73]. In the present study, all discriminating parameters were computed from the laboratory frame (*i.e.* the frame in which the vertical axis and horizontal plane are independent of the initial orientation of the sensor). Nevertheless, with the effect of the position of the sensor on the body segment, the CV was $< 5\%$ for our discriminating parameters, for realistic placement errors (we estimated our error as routine to be about 10 mm), and $< 10\%$ for extreme placement errors. Indeed, special care is needed for placement of the sensor, but this positioning had moderate impact on the parameters we propose.

We compared the IMU-based gait parameters and lower limb osteoarthritis assessed by the WOMAC index, which is a purely clinical score. Classically, inertial sensor based studies use the Kellgren and Lawrence radiographic score to rate knee osteoarthritis [5,26,27,53–57]. Radiographic knee osteoarthritis severity is known to have poor correlation with the clinic namely gait disturbance [74,75]. Radiographic osteoarthritis can be clinically silent [26], which could explain the inconsistent correlation between gait analysis and radiographic-based lower limb osteoarthritis severity [59]. Again, it can be hypothesized that pain commands walking strategies.

Finally, beyond the fact that we designed an automated method of gait quantification, adapted to daily practice, our results gave some insight in the impact of lower limb osteoarthritis on locomotion. The most relevant results of our study are the decrease of the mean and RMS of norm of the acceleration in the horizontal plane on both feet with disease severity. It could result from a diminution of movement in the AP direction due to pain. This interpretation had been suggested in studies relying on local peak amplitudes [13,54,76,77]. Liikavainio

et al. (2010) have also hypothesized that patients use a different strategy to brake the forward movement of the swinging leg before floor contact. This strategy could explain both the reduction of our global parameters and the increase of the local peaks in patients reported by others.

Conclusion

Our study showed that by using two IMUs placed on both feet and a signal processing method without step detection, we could objectively quantify limp in lower-limb osteoarthritis. This finding underlines the importance of measuring key anatomical landmarks and accessible gait parameters in exploring limp by using IMUs and severity grading. Although the proposed method still had some limitations, it provided new, automatically computed parameters that could be used for the comprehension of lower limb osteoarthritis in current medical practice. It may not only be used in medical consultation to score patients, but also to monitor the evolution of their clinical syndrome during and after rehabilitation. Finally, it paves the way for the quantification of gait in other fields such as neurology and for home monitoring.

Supporting Information

S1 Table. Lower limb osteoarthritis severity group, WOMAC score, BMI, age, walking velocity and the 60 parameters for the 12 control subjects and the 48 patients. Each parameter is defined by: a *sensor = {head*, *lower back*, *ipsilateral foot*, *contralateral foot}*; a *frame = {sensor*, *office}*; an *axis = {AP*,*ML*,*V}* if the frame is the sensor-linked frame or an *axis = {H*, *V}* if the frame is the doctor's office-linked frame (H for horizontal plane); a signal *sig = {acceleration*, *angular velocity}* and a statistical tool *stat = {mean*, *RMS}*. The parameter *ipsilateral foot-office-H-acceleration-mean-modified* corresponds to the parameter *ipsilateral foot-office-H-acceleration-mean* computed with the cumulative error for the manual exercise phase annotation that was estimated at 1 second. Accelerations are given in g and angular velocities in deg/s. AP antero-posterior, ML medio-lateral, V vertical, H horizontal plane, RMS root mean square, BMI body mass index.
(XLSX)

Acknowledgments

Blynn Shideler for writing assistance.

SATT Ile-de-France Innov for funding.

Author Contributions

Conceptualization: ThG ChL NV SB AY CdW PPV DR.

Data curation: RB AAP.

Formal analysis: RB LO ThM ChT AAP AV ChL.

Funding acquisition: PPV.

Investigation: RB ThG DR.

Methodology: ThG ChL NV SB AY CdW PPV DR.

Project administration: RB ThG PPV DR.

Resources: ThG.

Software: RB LO ThM ChT AAP AV ChL.

Supervision: ThG PPV DR.

Validation: RB ThG LO AAP AV ChL PPV DR.

Visualization: RB ThM DR.

Writing – original draft: RB LO PPV DR.

Writing – review & editing: RB LO SL PPV DR.

References

1. Childs JD, Sparto PJ, Fitzgerald GK, Bizzini M, Irrgang JJ. Alterations in lower extremity movement and muscle activation patterns in individuals with knee osteoarthritis. Clin Biomech 2004; 19:44–9.
2. McKean KA, Landry SC, Hubley-Kozey CL, Dunbar MJ, Stanish WD, Deluzio KJ. Gender differences exist in osteoarthritic gait. Clin Biomech 2007; 22:400–9. doi: 10.1016/j.clinbiomech.2006.11.006 PMID: 17239509
3. Astephen JL, Deluzio KJ, Caldwell GE, Dunbar MJ, Hubley-kozey CL. Gait and neuromuscular pattern changes are associated with differences in knee osteoarthritis severity levels. J Biomech 2008; 41:868–76. doi: 10.1016/j.jbiomech.2007.10.016 PMID: 18078943
4. Andriacchi TP, Hurwitz DE. Gait biomechanics and the evolution of total joint replacement. Gait Posture 1997; 5:256–64. doi: 10.1016/S0966-6362(97)00013-1
5. Tas S. Effects of severity of osteoarthritis on the temporospatial gait parameters in patients with knee osteoarthritis. ACTA Orthop Traumatol Turc 2014; 48:635–41. doi: 10.3944/AOTT.2014.13.0071 PMID: 25637727
6. Astephen JL, Deluzio KJ. Changes in frontal plane dynamics and the loading response phase of the gait cycle are characteristic of severe knee osteoarthritis application of a multidimensional analysis technique. Clin Biomech 2005; 20:209–17.
7. Brinkmann JR, Perry J. Rate and range of knee motion during ambulation in healthy and arthritic subjects. Phys Ther 1985; 65:1055–60. PMID: 4011684
8. Hanlon M, Anderson R. Prediction methods to account for the effect of gait speed on lower limb angular kinematics. Gait Posture 2006; 24:280–7. doi: 10.1016/j.gaitpost.2005.10.007 PMID: 16311035
9. Chen CPC, Chen MJL, Pei Y-C, Lew HL, Wong P-Y, Tang SFT. Sagittal plane loading response during gait in different age groups and in people with knee osteoarthritis. Am J Phys Med Rehabil 2003; 82:307–12. doi: 10.1097/01.PHM.0000056987.33630.56 PMID: 12649658
10. Stauffer RN, Chao EYS, Györy AN. Biomechanical gait analysis of the diseased knee joint. Clin Orthop Relat Res 1977; 126:246–55. PMID: 598127
11. Smith AJ, Lloyd DG, Wood DJ. Pre-surgery knee joint loading patterns during walking predict the presence and severity of anterior knee pain after total knee arthroplasty. J Orthop Res 2004; 22:260–6. doi: 10.1016/S0736-0266(03)00184-0 PMID: 15013083
12. Baliunas AJ, Hurwitz DE, Ryals AB, Karrar A, Case JP, Block JA, et al. Increased knee joint loads during walking are present in subjects with knee osteoarthritis. Osteoarthr Cartil 2002; 10:573–9. PMID: 12127838
13. Turcot K, Aissaoui R, Boivin K, Pelletier M, Hagemeister N, de Guise JA. New accelerometric method to discriminate between asymptomatic subjects and patients with medial knee osteoarthritis during 3-D gait. Biomed Eng IEEE Trans 2008; 55:1415–22.
14. Sharma L, Kapoor D, Issa S. Epidemiology of osteoarthritis: an update. Curr Opin Rheumatol 2006; 18:147–56. doi: 10.1097/01.bor.0000209426.84775.f8 PMID: 16462520
15. Herzog W, Federico S. Considerations on joint and articular cartilage mechanics. Biomech Model Mechanobiol 2006; 5:64–81. doi: 10.1007/s10237-006-0029-y PMID: 16534622
16. McConnell S, Kolopack P, Davis AM. The Western Ontario and McMaster Universities Osteoarthritis Index (WOMAC): A Review of Its Utility and Measurement Properties. Arthritis Care Res (Hoboken) 2001:453–61.
17. Roos EM, Klässbo M, Lohmander LS. WOMAC Osteoarthritis Index: Reliability, validity, and responsiveness in patients with arthroscopically assessed osteoarthritis. Scand J Rheumatol 1999; 28:210–5. PMID: 10503556
18. Bellamy N. Pain assessment in osteoarthritis: experience with the WOMAC osteoarthritis index. Semin Arthritis Rheum 1989; 18:14–7. PMID: 2786253

19. Angst F, Aeschlimann A, Steiner W, Stucki G. Responsiveness of the WOMAC osteoarthritis index as compared with the SF-36 in patients with osteoarthritis of the legs undergoing a comprehensive rehabilitation intervention. Ann Rheumatolgic Disord 2001:834–40.
20. Theiler R, Sangha O, Schaeren S, Michel BA, Tyndall A, Dick W, et al. Superior responsiveness of the pain and function sections of WOMAC Index as compared to the Lequesne-algofunctional Index in patients with osteoarthritis of the lower extremities. Osteoarthr Cartil 1999:515–9. doi: 10.1053/joca.1999.0262 PMID: 10558848
21. Witvrouw E, Victor J, Bellemans J, Rock B, Van Lummel R, Van Der Slikke R, et al. A correlation study of objective functionality and WOMAC in total knee arthroplasty. Knee Surgery, Sport Traumatol Arthrosc 2002; 10:347–51. doi: 10.1007/s00167-002-0302-2 PMID: 12444512
22. Lindemann U, Becker C, Muche R, Aminian K, Dejnabadi H, Nikolaus T, et al. Gait analysis and WOMAC are complementary in assessing functional outcome in total hip replacement. Clin Rehabil 2006; 20:413–20. PMID: 16774092
23. Hubble RP, Naughton G a., Silburn P a., Cole MH. Wearable Sensor Use for Assessing Standing Balance and Walking Stability in People with Parkinson's Disease: A Systematic Review. PLoS One 2015; 10:e0123705. doi: 10.1371/journal.pone.0123705 PMID: 25894561
24. Auvinet B, Berrut G, Touzard C, Moutel L, Collet N, Chaleil D, et al. Reference data for normal subjects obtained with an accelerometric device. Gait Posture 2002; 16:124–34. PMID: 12297254
25. Astephen Wilson JL. Challenges in dealing with walking speed in knee osteoarthritis gait analyses. Clin Biomech 2012; 27:210–2. doi: 10.1016/j.clinbiomech.2011.09.009 PMID: 22019141
26. Nagano Y, Naito K, Saho Y, Torii S, Ogata T, Nakazawa K, et al. Association between in vivo knee kinematics during gait and the severity of knee osteoarthritis. Knee 2012; 19:628–32. doi: 10.1016/j.knee.2011.11.002 PMID: 22192889
27. Kiss RM. Effect of severity of knee osteoarthritis on the variability of gait parameters. J Electromyogr Kinesiol 2011; 21:695–703. doi: 10.1016/j.jelekin.2011.07.011 PMID: 21840223
28. Mizuike C, Ohgi S, Morita S. Analysis of stroke patient walking dynamics using a tri-axial accelerometer. Gait Posture 2009; 30:60–4. doi: 10.1016/j.gaitpost.2009.02.017 PMID: 19349181
29. Van Den Noort JC, Van Der Esch M, Steultjens MPM, Dekker J, Schepers HM, Veltink PH, et al. The knee adduction moment measured with an instrumented force shoe in patients with knee osteoarthritis. J Biomech 2012; 45:281–8. doi: 10.1016/j.jbiomech.2011.10.027 PMID: 22079386
30. Freedman Silvernail J, Milner CE, Thompson D, Zhang S, Zhao X. The influence of body mass index and velocity on knee biomechanics during walking. Gait Posture 2013; 37:575–9. doi: 10.1016/j.gaitpost.2012.09.016 PMID: 23103243
31. Heiden TL, Lloyd DG, Ackland TR. Knee joint kinematics, kinetics and muscle co-contraction in knee osteoarthritis patient gait. Clin Biomech 2009; 24:833–41.
32. Jordan K, Challis JH, Newell KM. Walking speed influences on gait cycle variability. Gait Posture 2007; 26:128–34. doi: 10.1016/j.gaitpost.2006.08.010 PMID: 16982195
33. Landry SC, McKean KA, Hubley-Kozey CL, Stanish WD, Deluzio KJ. Knee biomechanics of moderate OA patients measured during gait at a self-selected and fast walking speed. J Biomech 2007; 40:1754–61. doi: 10.1016/j.jbiomech.2006.08.010 PMID: 17084845
34. Ying H, Silex C, Schnitzer a, Leonhardt S, Schiek M. Automatic Step Detection in the Accelerometer Signal. 4th Int Work Wearable Implant Body Sens Networks (BSN 2007) 2007; 13:80–5. doi: 10.1007/978-3-540-70994-7_14
35. Pan J, Tompkins WJ. A real-time QRS detection algorithm. IEEE Trans Biomed Eng 1985; 32:230–6. doi: 10.1109/TBME.1985.325532 PMID: 3997178
36. Marschollek M, Goevercin M, Wolf K-H, Song B, Gietzelt M, Haux R, et al. A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons. Eng. Med. Biol. Soc. 2008. EMBS 2008. 30th Annu. Int. Conf. IEEE, 2008, p. 1319–22.
37. Dijkstra B, Zijlstra W, Scherder E, Kamsma Y. Detection of walking periods and number of steps in older adults and patients with Parkinson's disease: Accuracy of a pedometer and an accelerometry-based method. Age Ageing 2008; 37:436–41. doi: 10.1093/ageing/afn097 PMID: 18487266
38. Fortune E, Lugade V, Morrow M, Kaufman K. Step Counts Using a Tri-Axial Accelerometer During Activity. AsbwebOrg n.d.:1–2.
39. Libby R. A simple method for reliable footstep detection on embedded sensor platforms. Sensors Peterbrgh NH 2008:1–16.
40. Thuer G, Verwimp T. Step Detection Algorithms for Accelerometers. E-Lab Master's Thesis, from Artesis Univ Coll Antwerp, Antwerp, Belgium 2009;2009:1–8.

41. Sejdić Ervin, Lowry KA, Bellanca J, Redfern MS, Brach JS. A comprehensive assessment of gait accelerometry signals in time, frequency and time-frequency domains. IEEE Trans Neural Syst Rehabil Eng 2015; 22:603–12. doi: 10.1109/TNSRE.2013.2265887.A

42. Din S Del, Godfrey A, Galna B, Lord S, Rochester L. Free-living gait characteristics in ageing and Parkinson ' s disease: impact of environment and ambulatory bout length. J Neuroeng Rehabil 2016:1–12. doi: 10.1186/s12984-016-0154-5 PMID: 27175731

43. Barrois R, Oudre L, Moreau T, Truong C, Vayatis N, Buffat S, et al. Quantify osteoarthritis gait at the doctor ' s office: A simple pelvis accelerometer based method independent from footwear and aging. Comput Methods Biomech Biomed Engin 2015:1880–1. doi: 10.1080/10255842.2015.1072414 PMID: 26315565

44. Moe-Nilssen R. A new method for evaluating motor control in gait under real-life environmental conditions. Part 1: The instrument. Clin Biomech 1998; 13:320–7.

45. Bolink SAAN, Lenguerrand E, Brunton LR, Wylde V, Gooberman-hill R, Heyligers IC, et al. Clinical Biomechanics Assessment of physical function following total hip arthroplasty: Inertial sensor based gait analysis is supplementary to patient-reported outcome measures. JCLB 2016; 32:171–9. doi: 10.1016/j.clinbiomech.2015.11.014 PMID: 26706048

46. Bolink SAAN, Brunton LR, Laarhoven S van, Lipperts M, Heyligers IC, Blom AW, et al. Frontal plane pelvic motion during gait captures hip osteoarthritis related disability. Hip Int 2016;0:0–0. doi: 10.5301/hipint.5000282 PMID: 26351120

47. Bolink SAAN, Grimm B, Heyligers IC. The Knee Patient-reported outcome measures versus inertial performance-based outcome measures: A prospective study in patients undergoing primary total knee arthroplasty. Knee 2015; 22:618–23. doi: 10.1016/j.knee.2015.04.002 PMID: 26032657

48. Staab W, Hottowitz R, Sohns C, Sohns JM, Gilbert F, Menke J, et al. Accelerometer and Gyroscope Based Gait Analysis Using Spectral Analysis of Patients with Osteoarthritis of the Knee. J Phys Ther Sci 2014; 26:997–1002. doi: 10.1589/jpts.26.997 PMID: 25140082

49. Madgwick SOH. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Rep X-Io Univ Bristol 2010: 32. doi: 10.1109/ICORR.2011.5975346

50. Cognolato M. Experimental validation of XSens inertial sensor during clinical and sport motion capture. 2012.

51. Astephen JL, Deluzio KJ, Caldwell GE, Dunbar MJ, Hubley-Kozey CL. Gait and neuromuscular pattern changes are associated with differences in knee osteoarthritis severity levels. J Biomech 2008; 41:868–76. doi: 10.1016/j.jbiomech.2007.10.016 PMID: 18078943

52. Brandes M, Schomaker R, Molenhoff G, Rosenbaum D. Quantity versus quality of gait and quality of life in patients with osteoarthritis. Gait Posture 2008; 28:74–9. doi: 10.1016/j.gaitpost.2007.10.004 PMID: 18054233

53. Huang SC, Wei IP, Chien HL, Wang TM, Liu YH, Chen HL, et al. Effects of severity of degeneration on gait patterns in patients with medial knee osteoarthritis. Med Eng Phys 2008; 30:997–1003. doi: 10.1016/j.medengphy.2008.02.006 PMID: 18417411

54. Liikavainio T, Bragge T, Hakkarainen M, Karjalainen PA, Arokoski JP. Gait and muscle activation changes in men with knee osteoarthritis. Knee 2010; 17:69–76. doi: 10.1016/j.knee.2009.05.003 PMID: 19556137

55. Mündermann A, Dyrby CO, Hurwitz DE, Sharma L, Andriacchi TP. Potential Strategies to Reduce Medial Compartment Loading in Patients With Knee Osteoarthritis of Varying Severity: Reduced Walking Speed. Arthritis Rheum 2004; 50:1172–8. doi: 10.1002/art.20132 PMID: 15077299

56. Mündermann A, Dyrby CO, Andriacchi TP. Secondary gait changes in patients with medial compartment knee osteoarthritis: increased load at the ankle, knee, and hip during walking. Arthritis Rheum 2005; 52:2835–44. doi: 10.1002/art.21262 PMID: 16145666

57. Zeni JA, Higginson JS. Differences in gait parameters between healthy subjects and persons with moderate and severe knee osteoarthritis: a result of altered walking speed? Clin Biomech 2009; 24:372–8.

58. Gök H, Ergin S, Yavuzer G. Kinetic and kinematic characteristics of gait in patients with medial knee arthrosis. Acta Orthop 2002; 73:647–52.

59. Heiden TL, Lloyd DG, Ackland TR. Knee joint kinematics, kinetics and muscle co-contraction in knee osteoarthritis patient gait. Clin Biomech 2009; 24:833–41. doi: 10.1016/j.clinbiomech.2009.08.005 PMID: 19765867

60. Hinman RS, Bennell KL, Metcalf BR, Crossley KM. Delayed onset of quadriceps activity and altered knee joint kinematics during stair stepping in individuals with knee osteoarthritis. Arch Phys Med Rehabil 2002; 83:1080–6. doi: 10.1053/apmr.2002.33068 PMID: 12161828

61. Turcot K, Aissaoui R, Boivin K, Hagemeister N, Pelletier M, de Guise JA. Test-Retest Reliability and Minimal Clinical Change Determination for 3-Dimensional Tibial and Femoral Accelerations During Treadmill Walking in Knee Osteoarthritis Patients. Arch Phys Med Rehabil 2008; 89:732–7. doi: 10.1016/j.apmr.2007.09.033 PMID: 18374005

62. Hirasaki E, Moore ST, Raphan T, Cohen B. Effects of walking velocity on vertical head and body movements during locomotion. Exp Brain Res 1999; 127:117–30. PMID: 10442403

63. Kaufman KR, Hughes C, Morrey BF, Morrey M, An K-N. Gait characteristics of patients with knee osteoarthritis. J Biomech 2001; 34:907–15. PMID: 11410174

64. Al-Zahrani KS, Bakheit AMO. A study of the gait characteristics of patients with chronic osteoarthritis of the knee. Disabil Rehabil 2002; 24:275–80. PMID: 12004973

65. Hurwitz DE, Ryals AB, Case JP, Block JA, Andriacchi TP. The knee adduction moment during gait in subjects with knee osteoarthritis is more closely correlated with static alignment than radiographic disease severity, toe out angle and pain. J Orthop Res 2002; 20:101–7. doi: 10.1016/S0736-0266(01)00081-X PMID: 11853076

66. Rutherford DJ, Hubley-Kozey CL, Stanish WD, Dunbar MJ. Neuromuscular alterations exist with knee osteoarthritis presence and severity despite walking velocity similarities. Clin Biomech 2011; 26:377–83. doi: 10.1016/j.clinbiomech.2010.11.018 PMID: 21185628

67. Zeni JA, Richards JG, Higginson JS. Two simple methods for determining gait events during treadmill and overground walking using kinematic data. Gait Posture 2008; 27:710–4. doi: 10.1016/j.gaitpost.2007.07.007 PMID: 17723303

68. Lewek MD, Rudolph KS, Snyder-Mackler L. Control of frontal plane knee laxity during gait in patients with medial compartment knee osteoarthritis. Osteoarthr Cartil 2004; 12:745–51. doi: 10.1016/j.joca.2004.05.005 PMID: 15325641

69. Kavanagh JJ, Menz HB. Accelerometry: a technique for quantifying movement patterns during walking. Gait Posture 2008; 28:1–15. doi: 10.1016/j.gaitpost.2007.10.010 PMID: 18178436

70. Moe-Nilssen R. Test-retest reliability of trunk accelerometry during standing and walking. Arch Phys Med Rehabil 1998; 79:1377–85. doi: 10.1016/S0003-9993(98)90231-3 PMID: 9821897

71. Sabatini AM. Wearable sensor systems in biomechanics: assessment of unrestrained walking features. Instrum. Meas. Technol. Conf. 2004. IMTC 04. Proc. 21st IEEE, vol. 2, 2004, p. 881–3.

72. Henriksen M, Lund H, Moe-Nilssen R, Bliddal H, Danneskiod-Samsøe B. Test—retest reliability of trunk accelerometric gait analysis. Gait Posture 2004; 19:288–97. doi: 10.1016/S0966-6362(03)00069-9 PMID: 15125918

73. Kavanagh JJ, Menz HB. Accelerometry: A technique for quantifying movement patterns during walking. Gait Posture 2008; 28:1–15. doi: 10.1016/j.gaitpost.2007.10.010 PMID: 18178436

74. Hannan MT, Felson DT, Pincus T. Analysis of the discordance between radiographic changes and knee pain in osteoarthritis of the knee. J Rheumatol 2000; 27:1513–7. PMID: 10852280

75. Dieppe PA. Relationship between symptoms and structural change in osteoarthritis. what are the important targets for osteoarthritis therapy? J Rheumatol 2004:50–3.

76. Lafortune MA. Three-dimensional acceleration of the tibia during walking and running. J Biomech 1991; 24:877–86. PMID: 1744146

77. Liikavainio T, Isolehto J, Helminen HJ, Perttunen J, Lepola V, Kiviranta I, et al. Loading and gait symmetry during level and stair walking in asymptomatic subjects with knee osteoarthritis: importance of quadriceps femoris in reducing impact force during heel strike? Knee 2007; 14:231–8. doi: 10.1016/j.knee.2007.03.001 PMID: 17451958

C

Template-based step detection from accelerometer signals

Template-based step detection from accelerometer signals

Laurent Oudre, Rémi Barrois-Müller, Thomas Moreau, Charles Truong, Stéphane Buffat, Pierre-Paul Vidal

***Abstract*—This article presents a method for step detection from accelerometer signals based on template matching. The principle of our step detection algorithm is to recognize the start and end times of the steps in the signal thanks to a predefined set of templates (library of steps). The algorithm is tested on a database of 1020 recordings, composed of healthy patients and patients with various neurological or orthopaedic troubles. Simulations on more than 40000 steps show that even with a library of only 5 templates, our method achieves remarkable results with a 98% recall and a 98% precision. The method is robust to parameter changes, adapts well to pathological subjects and can be used in a medical context for robust step estimation and gait characterization.**

***Index Terms*—gait analysis, biomedical signal processing, pattern recognition, step detection, physiological signals**

I. Introduction

PATHOLOGIES affecting posture, balance, and gait control are threatening the autonomy of patients not to mention the risk of fall and therefore require rehabilitation intervention as early as possible. However, it remains difficult to accurately evaluate the various specific interventions during the rehabilitation process and the optimal content of exercise interventions they should involve. If only for these reasons, it would be interesting to learn how to monitor motor sensorimotor behavior at large and locomotion in particular which is a growing area in medical engineering science [1], [2], [3], [4], [5], [6], [7]. It requires several steps: first, we wish to investigate how to monitor sensorimotor processing in behaving patients in the doctor office and the resulting cognitive load it implies. Second, we want to learn how to construct databases with the quantitative variables recorded in that process, in order to make longitudinal studies of behaving individuals. Third, we would like to merge these individual databases in large data banks to define statistical norms, which is mandatory to detect dysfunctions or pathologies at the earliest stage possible. In that process we meet at least three main problems: using pervasive or ubiquitous computing to collect data; facing large inter-individual variability in the studied HMCs; aggregating highly heterogeneous data to build the databank.

There exist many software applications on the market that use wearable sensors (namely accelerometers, gyroscopes, magnetometers and/or GPS) to calculate the number of steps made in a day [8], [9], the traveled distance in a day [10], [11], the average speed, the daily amount of time spent in walking, running, sitting, standing, laying [12], [13], useful for rehabilitation. Most of the algorithms published in this context are either dedicated to one specific terminal or mobile phone, or they are copyrighted and not freely available for research.

The main idea behind the algorithm presented in this paper is to automatically detect the steps from inertial sensor signals thanks to a library of templates extracted from real signals. It provides a novel, robust and precise step detection method which allows the user not only to count the steps, but also to locate when they occurred, how long they lasted, etc. These features can be useful either for personal or medical use. In particular, the algorithm has been tested on a large database containing 1020 walk exercises performed by healthy and pathological subjects at unconstrained speeds, which confirms the robustness of the presented method.

This article is organized as follows: Section II defines the task of step detection and gives an overview of state-of-the-art methods. Section III describes the data used for training and testing, the method, and the evaluation metrics. Section IV presents the results of our method, the influence of the parameters and compares the algorithm to state-of-the art methods. Section V provides a discussion on the robustness of the method and several insights for the possible use of this algorithm in a clinical context.

II. Background

A. What is a step ?

Locomotion is a hierarchical and complex phenomenon composed of different entities such as strides, steps, and phases [14], [1].

- Considering one foot, the stride is the succession of two phases: the *swing phase* (when the foot is in the air), and the *stance phase* (when the foot is in contact with the ground). The stance phase occurs between the heel-strike (moment when the foot hits the ground) and the toe-off (moment when the toes go off the ground), while the swing phase occurs between the toe-off and the next heel-strike.
- A *stride* is defined as the event that occurs between two heel-strikes of the same foot.
- A *step* is defined as the event that occurs between successive heel strikes of opposite feet. A stride is therefore composed of two steps: one for the right foot, one for the left foot.

L. Oudre is with L2TI, Université Paris 13, France and with COGNAC-G (UMR 8257), CNRS University Paris Descartes, France.

R. Barrois-Müller, Stéphane Buffat and P.P. Vidal are with COGNAC-G (UMR 8257), CNRS Université Paris Descartes, France.

T. Moreau and C. Truong are with CMLA (UMR 8536), CNRS ENS Cachan, France and with COGNAC-G (UMR 8257), CNRS Université Paris Descartes, France.

Manuscript received April 19, 2005; revised September 17, 2014.

In the formal medical definition, a step is supposed to start when the heel strikes the ground and to finish somewhere in the end of the stance phase. It is not related to the foot activity since the foot is also moving in the swing phase. We choose in this article another definition: a step is defined in the following as the whole period of activity of a foot (when the foot is moving). The beginning of the step is defined as the heel-off (moment when the heel leaves the floor) and end of the step is defined as the foot-flat (moment when the foot is stabilized on the floor).This new definition allows to consider the whole period of activity of a foot as a step, which makes it more adapted to step detection. Note that it does not change the number of steps and that it is easy to switch back to the medical definition once the heel-off and foot-flat instants have been detected.

B. Existing methods

Current algorithms can be classified in two categories:

- Step counting algorithms: the aim is only to know the number of steps performed by the subject
- Step detection algorithms: the aim is to locate when the step occurred, and eventually to give specific timings (heel-strike, toe-off, etc.). These algorithms can also be used for step counting.

Among step detection algorithms, two main approaches have been proposed: the use of filtering/thresholding/peak detection techniques and the use of template matching. The former aims to recognize one specific event, supposedly characteristic of the step (such as a local maximum or the time when the signal exceeds a threshold). Most of the time, these algorithms include a preprocessing step where the signal is filtered so as to emphasize the event that they seek to detect or to remove other events. The most well-known pre-processing stage was designed by Pan-Tompkins [15] and is composed of several signal processing blocks (bandpass filtering, derivation, squaring, etc.). Designed at first for ECG signals, this pre-precessing has been used in various step detection methods [16], [17], [2], [18]. After this possible processing stage, the steps are detected with empirical or dynamic thresholds, peak detection methods, of a combination of both [19], [4], [20]. Other methods seek to detect each phase of the walking process by using dedicated signal processing techniques (such as peak detection, zero-crossing, etc.) [3], [5]. Unfortunately, these methods heavily rely on the calibration of several parameters (width of the bandpass filter, window length, thresholds, etc.) [16], [17], [2], [18] which are difficult to estimate and thus set according to empirical experience. Moreover, these methods often assume some prior knowledge on the shape of a step [3], [5], which significantly limits the detection of unconventional patterns found with mobility-impaired patients.

For these reasons, we have decided in this article to focus on the second type of step detection methods, based on template matching. The main intuition behind this is that there are several types of steps (according to interpersonal variability, age, speed and pathology). Therefore, it is irrelevant to try to detect steps with one specific model (which is basically what is done with other methods since they only consider one set of parameters, thresholds, detection criteria, etc.). In order to overcome this issue, it is necessary to use a library of models (in our case a library of patterns) which represent typical step cycles. Hopefully, the use of this library can improve the robustness of the detection and paradoxically, prevent the overfitting induced by the choice of many parameters. Note that while commonly used in several other fields, this approach is novel in the context of step detection. We are aware of only one article mentioning the use of templates for step detection [16] is using one single template automatically extracted with filtering/thresholding/peak detection methods (thus relying on many parameters) and not from raw data. Also, in their paper, a different template is extracted for each subject, and only used for this particular subject. The novelty of the algorithm presented in this paper is that it uses a limited set of parameters whose influence is carefully studied and analysed. Also, our method is tested on a large database, with healthy and pathological subjects, at various speeds and in a rigorous cross-validation context.

III. Data, method and evaluation

A. Data acquisition and first observations

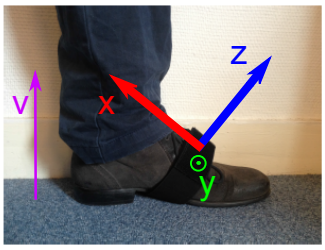
The data used for the conception and testing of the method presented in the article has been provided by the following medical departments: Service de chirurgie orthopédique et de traumatologie de l'Hôpital Européen Georges Pompidou, Assistance Publique des Hôpitaux de Paris, Service de médecine physique et de réadaptation de l'Hôpital Fernand Widal, Assistance Publique des Hôpitaux de Paris, Service de neurologie de l'Hôpital dInstruction des Armées du Val de Grâce, Service de Santé des Armées. The study was validated by a local ethic comity (Comité de Protection des Personnes Ile de France II, CPP 2014-10-04 RNI) and both patients and control subjects gave their written consent to participate. All signals have been acquired at 100 Hz with wireless XSens MTw™ sensors located at the right and left foot and fixed using a velcro band designed by XSens™. The signals obtained with both sensors were automatically synchronized by the acquisition software. All subjects were asked to:

- stand quiet for 6 seconds
- walk 10 meters at preferred walking speed on a level surface
- make a U turn
- walk back
- stand quiet 2 seconds

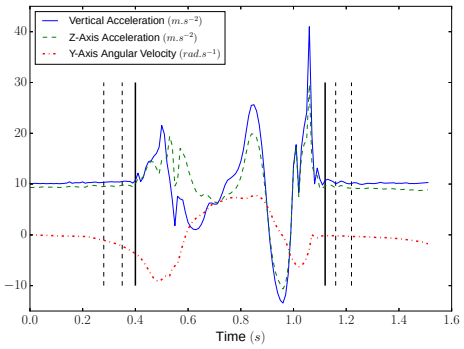
For practical reasons, patients kept their own shoes. The database is composed of 230 subjects who performed the protocol between 1 and 10 times, which leads to 1020 recordings. The subject's characteristics are presented in Table I. Healthy subjects had no known medical impairment. The orthopedic group is composed of 2 cohorts of distinct pathologies: lower limb osteoarthrosis and cruciate ligament injury. The neurologic group is composed of 4 cohorts: hemispheric stroke, Parkinsons disease, toxic peripheral neuropathy and radiation induced leukoencephalopathy.

Group	Number of exercises	Number of subjects	Sex (M/F)	Age (yr)	Height (cm)	Weight (kg)
Healthy subjects	242	52	35/17	36.4 (20.6)	173.4 (10.8)	70.7 (12.2)
Orthopedic diseases	243	53	26/27	60.1 (19.3)	169.2 (10.2)	77.4 (16.8)
Neurologic diseases	535	125	80/45	61.6 (13.2)	169.8 (8.7)	72.7 (15.5)
Total	1020	230	141/89	55.5 (19.6)	170.5 (9.7)	73.4 (15.3)

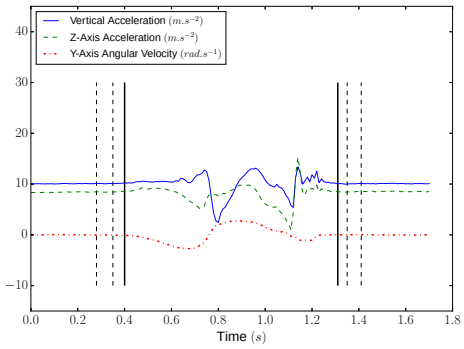
TABLE I: Subjects' characteristics. For the age, height and weight, the mean and the standard deviations are displayed.



(a) Definition of the axis for the XSens™ sensor located at the left foot



(b) Healthy patient



(c) Hip affected patient

Fig. 1: (a) XSens™ sensor - (b,c) Vertical acceleration, Z-axis acceleration and the Y-axis angular velocity recorded from the right foot. The vertical lines displays the different possibilities for start/end times.

The protocol includes 2 sensors (left and right foot), and each of them records a 9-dimensional signal (3D accelerations, 3D angular velocities, 3D magnetic fields), possibly with some recalibrated data provided by the XSens™ software (such as the vertical acceleration in the direction of the gravity). Instead of considering all these dimensions, we decided to only use a subset of them, and select the most relevant in the context of step detection. This decision has been made based on observations of real data and physiological reasons provided by doctors. We decided to only select the components that are the most reflective of the locomotion process (see Figure 1a for the definition of the axis): the Z-axis acceleration, the recalibrated vertical acceleration (vertical movements of the foot) and the Y-axis angular velocity (swing in the direction of the walk). We expect these components to strongly react to the steps, making them identifiable.

Examples of these 3 components (Z-axis acceleration, vertical acceleration and Y-axis angular velocity) recorded at the right foot are presented on Figures 1b and 1c for respectively an healthy and hip-injured patient. It appears on these figures that the amplitudes of the signals are clearly different and it is likely that classical threshold-based methods would hardly perform well on both subjects. However, the structure and shape of the step is roughly the same for both subjects so it might be relevant to use a template-base method. Nevertheless, these examples also display the main difficulties in conceiving an automatic algorithm for step detection:

- The uncertainties in the definition of the starts and ends of the steps. Indeed, we can see on Figure 1b, that many choices would be acceptable: depending on the considered definition, the results may be different.
- The variability of the step patterns according to the pathology, the age, the weight, etc. For example, on Figure 1c, the subject is dragging his feet, causing an abrupt change in the step pattern (noisy part at the end of the step).

B. Description of the method

The principle of our step detection algorithm is to recognize the steps in the signals thanks to a predefined set of templates. More precisely, our method uses a set of templates $\mathcal{P}$: these templates have been manually extracted from real accelerometer data and checked by doctors and specialists of locomotion. Each template $p \in \mathcal{P}$ is a three-dimensional signal of length $|p|$ (vertical acceleration, Z-axis acceleration and Y-axis angular velocity) corresponding to one step.

These templates are to be compared to the signal we want to study by calculating some correlation coefficients. As the sequences we want to detect are variable in duration as well as in amplitude, we want to use a measure of fit that is independent of the scale but is able to identify the correspondences in shape. Also, we want the comparison to be independent of the orientation of the sensor, so any DC component should be removed. In this context, it seems natural to use the Pearson correlation coefficient, which satisfies all these conditions, and defined for two one-dimensional vectors y and z of length n as

$$\rho_{y,z} = \frac{\mathrm{cov}(y,z)}{\sigma_y \sigma_z} = \frac{E[(y-\mu_y)(z-\mu_z)]}{\sigma_y \sigma_z} \qquad (1)$$

where (μ_y, μ_z), (σ_y, σ_z) are respectively the mean and standard deviation of y and z.

Let x be a three-dimensional signal: we want to detect the steps by using the set of templates $\mathcal{P}$. Let us introduce the following notations:

- $|\mathcal{P}|$ is the number of three-dimensional templates
- $|x|$ (resp. $|p|$) is the length of the three-dimensional vector x (resp. p)
- $x^{(k)}$ (resp. $p^{(k)}$) is the k^{th} component of x (resp. p). In our case we have $k \in \{1, 2, 3\}$
- $x^{(k)}[t_1 : t_1]$ is the portion of $x^{(k)}$ between time samples t_1 and t_2 (we therefore have $x^{(k)}[1 : |x|] = x^{(k)}$)

The first step of the algorithm consists in calculating the Pearson correlation coefficients between the templates and the signal, for all possible time positions and all three components:

$$\begin{aligned} \forall k \in \{1,2,3\}, \quad \forall p \in \mathcal{P}, \quad \forall t \in [\![1, |x| - |p| + 1]\!] \\ r(k,p,t) = \rho\left(p^{(k)}, x^{(k)}[t : t + |p| - 1]\right) \end{aligned} \tag{2}$$

$r(k, p, t)$ is the correlation between the k^{th} component of template p and the k^{th} component of the signal at time sample t.

The second step is a local maxima search among the $r(k, p, t)$ coefficients in order to extract the possible steps. $r(k, p, t)$ is selected as a local maximum if it is greater than its nearest temporal neighbors. We define the set $\mathcal{L}$ of possible steps as:

$$\begin{aligned} \mathcal{L} = \{(k,p,t) \text{ s.t. } r(k,p,t) > r(k,p,t-1) \\ \text{and } r(k,p,t) > r(k,p,t+1)\} \end{aligned} \tag{3}$$

The $\mathcal{L}$ contains all acceptable positions for the steps, and the coefficients $r(k, p, t)$ with $(k, p, t) \in \mathcal{L}$ can be interpreted as the likelihood of having a step similar to the pattern p at time sample t.

Our step detection algorithm takes as input the set $\mathcal{L}$ and works as a greedy process. At each iteration, the largest value $r(k^*, p^*, t^*)$ with $(k^*, p^*, t^*) \in \mathcal{L}$ is selected: if the step p^* positioned at time sample t^* overlaps with a previously detected step, it is discarded and we switch to the next largest value. Otherwise, if step p^* can be positioned at time t^*, the step is detected and all time samples between t^* and $t^* + |p^*| - 1$ are forbidden for the next iterations. The process is stopped when all time samples are forbidden, when the set of possible steps $\mathcal{L}$ is empty, or when all values $r(k, p, t)$ with $(k, p, t) \in \mathcal{L}$ are lower than a threshold λ. Note that in practice, the main purpose of threshold λ is to speed up the algorithm, as it reduces the size of set $\mathcal{L}$. The algorithm is summarized on Algorithm 1.

A last post-processing step can be performed so as to discard the steps detected when the patient was actually not moving. These false detections occur when a fit is found with one template, even though the signal is almost equal to zero after DC component removal: this is in fact due to the invariance in scale provided by the Pearson correlation coefficients. A solution can be found by processing the final list of detected steps, and removing the steps whose standard deviation is way lower than the one of the template that was used for the detection. Formally, this step involves a threshold μ: given a detected step with start and end times t_{start} and t_{end}, detected thanks to the pattern $p^{(k)}$, the step is to be discarded if

$$\sigma_{x^{(k)}[t_{start}:t_{end}]} < \mu \ \sigma_{p^{(k)}} \tag{4}$$

where $\sigma_{\cdot}$ stands for the empirical standard deviation operator.

Algorithm 1: Step detection algorithm

Input: Set of possible steps $\mathcal{L}$
Output: Set of start times $\mathcal{T}_{start}$, set of end times $\mathcal{T}_{end}$

```
Set of forbidden time positions F = ∅;
T_start = ∅, T_end = ∅;

while F ≠ {1, ..., |x|} or L ≠ ∅ or max L > λ do
    (k*, p*, t*) = argmax_{(k,p,t)∈L} r(k, p, t);
    if {t*, ..., t* + |p*| − 1} ∉ F then
        t* → T_start;
        t* + |p*| − 1 → T_end;
        {t*, ..., t* + |p*| − 1} → F;
    end
    L = L\(k*, p*, t*);
end
```

C. Evaluation

All steps were manually annotated by specialists using a software allowing to point with the mouse the starts (foot-flat) and the ends (heel-off) of the foot flat periods during which the sensor is not moving. The annotations were performed thanks to the Z-axis acceleration (normal to the upper foot surface) which is the most sensitive direction to detect the movements of the foot with respect to the floor. For the tricky cases of pathological gaits, a first gross annotation was made and then refined by zooming on each step. The uncertainty of this annotation is evaluated to less than 0.2 s (20 samples) for each mouse click. In total, the database is composed of 40453 steps (20233 extracted on the right foot and 20220 on the left foot). Even though they had a distinct shape, the U-turn steps were also taken into account.

The following precision/recall metrics are used for the evaluation of our method based on the annotations provided by the specialists.

Precision. A detected step is counted as correct if the mean of its start and end times lies inside an annotated step. An annotated step can only be detected one time. If several detected steps correspond to the same annotated step, all but one are considered as false. The precision is the number of correctly detected steps divided by the total number of detected steps.

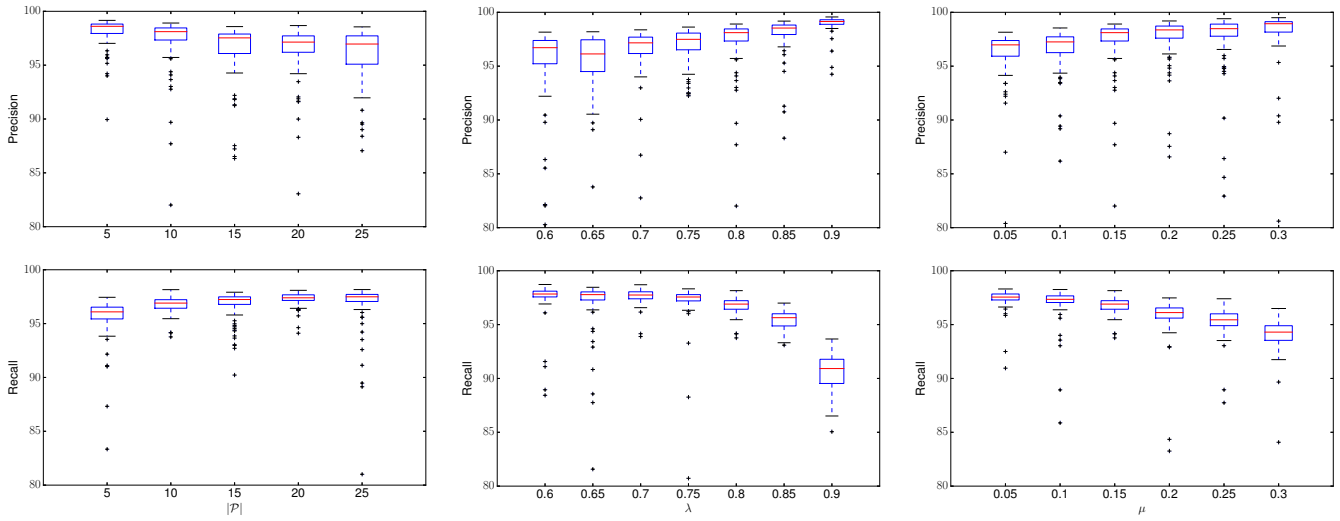
Recall. An annotated step is counted as detected if the mean of its start and end times lies inside a detected step. A detected step can only be used to detect one annotated step. If several annotated steps are detected with the same detected step, all but one are considered undetected. The recall is the number of detected annotated step divided by the total number of annotated steps.

IV. Results

A. Influence of the parameters

The algorithm depends on 3 numerical parameters:

- The size of the pattern library $|\mathcal{P}|$



(a) Influence of $|\mathcal{P}|$ on the precision and recall (in %)

(b) Influence of λ on the precision and recall (in %)

(c) Influence of μ on the precision and recall (in %)

Fig. 2: Influence of the parameters (on 100 simulations). By default, $|\mathcal{P}| = 10$, $\lambda = 0.8$ and $\mu = 0.15$. Boxes correspond to quartiles and median, whiskers to 5 and 95 percentiles. Outliers are represented as +

- The stopping criterion λ
- The threshold for discarding periods of no activity μ

Note that the algorithm is also influenced by the choice of the templates composing the library $\mathcal{P}$: this will be studied in the next section.

In order to study the scope of influence of these 3 parameters, a cross validation process is used:

- $|\mathcal{P}|$ three-dimensional step patterns are randomly chosen, so as to form the pattern library $\mathcal{P}$
- In order to avoid overfitting, all exercises performed by subjects that are used in the pattern library are then discarded from the test database.
- For each exercise of the test database, the step detection is performed with the $|\mathcal{P}|$ templates, and the detected steps are compared to the annotations

For each simulation, the mean and standard deviation of the precision/recall scores on the test database are calculated, as described in section III-C. This process is performed 100 times.

The parameters are studied with the following grid search:

- $|\mathcal{P}| : [5, 10, 15, 20, 25]$
- $\lambda : [0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9]$
- $\mu : [0.05, 0.1, 0.15, 0.2, 0.25, 0.3]$

In total, 210 different configurations are considered.

The configuration giving the best average results on 100 simulations is using $|\mathcal{P}| = 10$ templates, $\lambda = 0.8$ and $\mu = 0.15$, with an average recall of 96.59% (std: 4.91) and an average precision of 97.03% (std: 3.69). Note that these values correspond to the average on 100 simulations with randomly chosen templates: they do not reflect the optimal performances of the algorithm.

We propose to use this configuration as a reference and study the influence of the parameters from this grid node. Figure 2 presents the influence of the parameters on the precision and recall: on each figure, two of the parameters are fixed while the last one varies. The plots displays as boxplots the results obtained on the 100 simulations corresponding to the considered configuration.

On Figure 2a, it is visible that adding more templates to the library tends to increase the recall, but it has a negative effect on the precision. This is probably due to the cross-validation process used for testing. Since the templates are randomly chosen, it is unknown if they belong to healthy or pathological subjects, to forward walking or U-turn, etc. Therefore, when $|\mathcal{P}|$ increases, it also increases the probability that a pathological step is used for detection. This is one of the predictable effect of this experiment: if a step within the library is *unadapted* for the task, it causes false detection and thus lowers the performances. However, this does not mean that adding *appropriate* steps in the library would degrade the performances: this problem will be investigated in the next section (as well as the questionable notion of *appropriate steps*). When $|\mathcal{P}| = 5$, the limits of the algorithm are reached: due to the small number of templates, the method crucially depends on the choice of the templates used for detection, thus causing a large number of outliers. The best compromise between precision and recall is obtained for $|\mathcal{P}| = 10$, but this might only be due to the cross-validation setting: rather than an optimal number of templates to be used, it is likely that the composition of the library is more crucial to the performances of the algorithm.

The plot on Figure 2b is coherent with the definition of the parameter: when λ increases, only steps that are very correlated to the templates are selected: this increases the precision, but decreases the recall. On the contrary, when λ decreases, all possible steps are considered: the recall increases and the precision decreases. These results also confirm the utility of parameter λ: by increasing λ to an appropriate

value (around 0.6-0.8), it is possible to increase the precision (and the robustness of the precision) while keeping the recall constant. Also, λ has an impact on the computational cost: for example, using $\lambda = 0.8$ instead of $\lambda = 0$ allows to compute the results approximately 2 times faster. It is therefore interesting to use the largest value of λ as possible. The best average performances are obtained for $\lambda = 0.8$, which constitutes a good compromise between recall and precision: indeed, with $\lambda = 0.85$ some annotated steps are discarded and the recall drops.

Figure 2c shows that parameter μ mainly influences the recall. Indeed, when μ is too large, all steps whose amplitude are too different from those of the templates are discarded. This has a double effect: if one of the templates corresponds to a pathological patient whose steps have small amplitude, then it will not be able to detect steps on healthy patients. The opposite situation can also occur. In fact, when μ increases, the normalization effect provided by the Pearson correlation coefficient (1) is neutralized. Figure 2c shows that μ should be no greater than 0.2 so that the recall does not drop.

B. Influence of the composition of the library

The performances of the algorithm are intuitively dependent of the library of templates used for detection. As previously seen, when inappropriate steps are added to the library, the performances may drop. What would happen if the library of templates is composed only of healthy steps, but is to be used on patients with degraded walking abilities ? In order to correctly detect steps for a patient having e.g. an orthopedics disease, is it necessary to have patients with similar pathologies in the library of templates ?

To investigate this question, we propose to define two classes of subjects within the database: class A is typically composed of subjects who have no problem for walking, and class B is composed of subjects with severe pathologies that critically affect their locomotion. The idea is to study the cross-performances of the method on these two classes. The definition of these classes are non-trivial since the database contains gait recordings of patients cared for lower limb osteoarthritis, anterior cruciate ligament injury, hemispheric stroke, Parkinsons disease and neuropathy. In each nosologic class, patients were quoted by the medical doctors of our group with clinical scales specific to each pathology (WOMAC index : lower limb osteoarthritis ; Tegner Lysholm Knee Scoring Scale : anterior cruciate ligament injury ; Lower Limb Fugel Meyer scale : stroke ; UPDRS III : Parkinsons Disease ; TNSc : neuropathy). To allow the between pathology comparison, a transversal walking score (between 0 and 4) was assigned to each patient by the medical doctors of our group. Subjects with no problem for walking were graded 0, while other were graded from 1 to 4 (4 being the most severe degradation of locomotion). To have an idea, lower limb osteoarthritis patients with high functional manifestation walking troubles (use of cane, unable to climb stairs) were graded 4. Class A is defined as subjects with a locomotion grade of 0 (no problem) and Class B as subjects with locomotion grade of 3 or 4. In total 116 subjects are isolated from the database: 72 subjects in Class A (322 exercises, 4877 left steps, 4846 right steps), and 35 subjects in Class B (111 exercises, 3554 left steps, 3567 right steps).

In each simulation, the library is composed of templates belonging to only one class, and the test is performed on exercises belonging to only one class. All simulations are run with the default parameters $|\mathcal{P}| = 10$, $\lambda = 0.8$ and $\mu = 0.15$ (that gave the best average performances on 100 simulations in the grid search). Table II presents the results (recall/precision) averaged on 100 simulations. A first observation is that Class A and Class B templates give similar (and good) performances on Class A subjects. This confirms the intuitive idea that it is easier to detect steps for healthy subjects. However, Class B templates used on Class B subjects do not perform so well: it might be due to the definition of the class which involves several types of pathologies. In fact, these severe pathologies might affect the steps shapes in a different way, so even though some pathological templates are used for detection, they might not correspond to the particular pathology of the test subject. To increase the scores, two strategies can be implemented: either introduce all types of degradations within the library, or add several healthy (or less pathological) steps which could smooth the results by introducing less specific examples. Interestingly, the results obtained on Class B subjects with random templates and with the exact same parameters (see Section IV-A) are better than those obtained by using only Class B templates. This tends to show that in order to detect steps on severe pathological subjects, it is necessary to use a library composed of both healthy (or slightly pathological) and pathological steps.

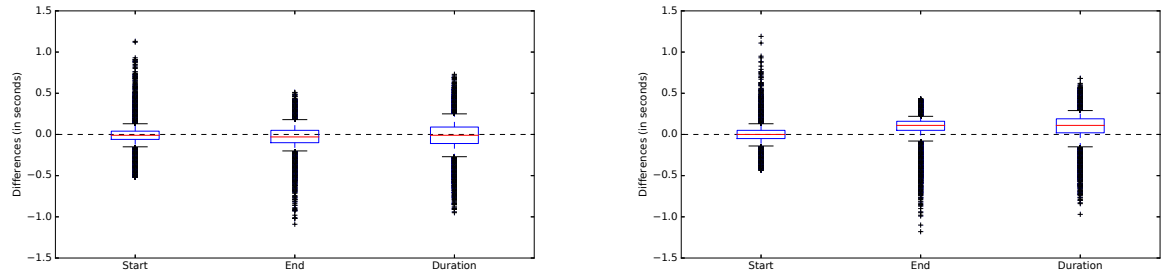
As far as cross-class detection is concerned, it seems that using only Class A templates for detecting Class B steps is not appropriate : the recall drops while the precision decreases. It is likely that these results are due to the amplitudes of the steps that greatly vary between healthy and pathological subjects. Due to parameter μ, steps with low amplitude are hardly detectable with high amplitude templates (and vice-versa). Also, the durations of the steps might be inappropriate for detection, since pathological steps are in general longer than healthy steps.

To summarize, two trends can be identified: as far as healthy subjects are concerned, the choice of templates is not crucial for the detection. But if the algorithm is to be used on pathological subjects, it appears that the best compromise would be to use a combination of healthy and pathological templates.

C. Detailed results for the best simulation

The best simulation on the whole grid search (21000 simulations) described in Section IV-A is using parameters $|\mathcal{P}| = 5$, $\lambda = 0.75$ and $\mu = 0.1$, with 98.40% recall and 98.44% precision. In this section, we propose a detailed study of this method. Note that this particular method should only be seen as a good association (templates + λ + μ) performing well, and does not constitute a golden standard (similar scores are obtained on several other simulations).

The detailed performances of this method on the whole database is presented on Table III: it is noticeable that scores



(a) Best step detection method ($|\mathcal{P}| = 5$, $\lambda = 0.75$ and $\mu = 0.1$)

(b) Best step detection method with one template ($|\mathcal{P}| = 1$, $\lambda = 0.6$ and $\mu = 0.15$)

Fig. 3: Differences between detected and annotated times (start, end and duration) for the best step detection method and the best step detection method with one template. Boxes correspond to quartiles and median, whiskers to 5 and 95 percentiles. Outliers are represented as +.

		Test data	
		Class A	Class B
Template data	Class A	R : 97.64 (1.17) P : 97.45 (4.46)	R : 89.74 (3.82) P : 95.75 (5.09)
	Class B	R : 97.80 (1.32) P : 97.28 (2.17)	R : 93.25 (4.17) P : 93.13 (5.76)

TABLE II: Influence of the composition of the library of templates in the step detection ($|\mathcal{P}| = 10$, $\lambda = 0.8$ and $\mu = 0.15$). Average recall and precision on 100 simulations (with standard deviation). Class A: subjects who have no problem for walking. Class B: subjects with severe pathologies that critically affect their locomotion.

are consistent on all groups of subjects. The best performances are obtained for healthy subjects, but there is no significant differences between the groups. This clearly shows that the method adapts well to different types of pathologies.

Out of the 40344 detected steps, 85% of them were detected with the Y-axis angular velocity, 2% with the vertical acceleration and 13% with Z-axis acceleration. This proportion can be due to the nature of the signals: medio-lateral angular velocity is actually known to be the direction in which there is the greatest quantity of movement during walking. This signal is often used in step detection [21], [22], and it is likely that this component captures a locomotion pattern that is the most reproducible among the subjects.

The good performances of this method are intuitively linked to the templates composing the library. It is remarkable that this method only requires a small number of templates, which tends to show that the algorithm do not need a large library to perform accurately. It probably rather needs a carefully selected set of templates, that are generic enough to fit the general shape of a step, but can also adapt to pathological steps. For instance, this library of 5 templates is composed as follows: 1 step belonging to an healthy subject, 3 steps corresponding to neurological diseases (2 with moderate troubles and 1 with severe troubles), and 1 step associated to orthopedic diseases (with moderate troubles). This covers all groups of subjects and the proportion of each group in the library is similar to the one of database. In particular, the neurological group is composed of many different diseases and it is likely that several patterns are necessary to accurately fit the whole range of step shapes.

In order to further investigate the accuracy of the method, some additional evaluation metrics are computed. For all correctly detected steps, we compute:

- the difference between the detected start time and the annotated start time
- the difference between the detected end time and the annotated end time
- the difference between the duration of the detected step and the duration of the annotated step

The repartition of these metrics on all 39677 correctly detected steps are presented on Figure 3a. One interesting result is that our method does not introduce a bias: the median of the differences for all times (start, end, duration) is approximately equal to zero, and the quartiles are symmetric. This tend to prove that the library is able to accurately detect the step boundaries and to adapt to various step durations. For 90% of the steps (represented as whiskers on the figure), the errors for start, end and duration times are lower than 0.25 seconds (in absolute value), which corresponds to 25 samples. These results are satisfactory when compared to the annotations uncertainties of experts and specialists (which are around 20 samples - see Section III-C). Outliers are in fact due to two specificities of the database: tiny steps (under 50 samples) mainly located during U-turn (causing underestimation for start times and overestimation of end and duration times), and highly pathological steps for stroke subjects whose duration exceeds one second (causing upper outliers for start times and lower outliers on end and duration times). The method tested here is using five templates of durations 65, 76, 82, 86 and 105 samples and the detection is inevitably constrained by these step durations. While this phenomenon does not penalize the results on most steps, it is one limit of the algorithm especially with small libraries. Should these outliers become more frequent, one possible solution is to increase the number of templates and to add typical steps corresponding to these outliers within the library.

	Best simulation		Pan-Tomkins		One template	
Group	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**
Healthy subjects	98.93 (2.22)	98.98 (2.43)	99.14 (1.71)	97.09 (3.60)	99.03 (2.06)	99.33 (1.76)
Orthopedic diseases	97.54 (2.92)	98.77 (2.12)	98.78 (2.09)	94.87 (5.09)	97.37 (3.06)	98.85 (2.23)
Neurological diseases	98.55 (3.05)	98.05 (3.02)	96.80 (3.52)	95.49 (4.55)	98.11 (3.31)	98.58 (2.55)
Total	98.40 (2.89)	98.44 (2.72)	97.82 (3.07)	95.72 (4.56)	98.15 (3.05)	98.82 (2.33)

TABLE III: Detailed performances of the best step detection method ($|\mathcal{P}| = 5$, $\lambda = 0.75$ and $\mu = 0.1$), the best Pan-Tomkins method, and the best step detection method with one template ($|\mathcal{P}| = 1$, $\lambda = 0.6$ and $\mu = 0.15$). Means and standard deviations are displayed.

D. Comparison with the state-of-the-art

The reference procedure for step counting/detection is based on the Pan-Tomkins method [15]. First intended for ECGs, it was later adapted to detect steps in the vertical accelerometer signal [16], [17], [2], [18]. It is composed of several successive signal processing steps, which are designed to emphasize the structure of the step, making it easier to detect. These steps can be summarized as:

- Bandpass filtering (between f_{min} andf_{max}): removes the gravity component and the noise.
- Derivation: amplifies the slope changes in the filtered signal. Whenever the foot rises from the ground or the heel hits the ground, the acceleration slope changes significantly and it translates into a burst in the filtered signal.
- Squaring: makes all points positive and enhances the large values of the filtered signal.
- Integration: the signal is smoothed using a moving-window integrator of length N_{inte}.
- Peak search procedure: originally, [15] used a threshold to find the phenomena they were looking for in the heart rate signal (every time the filtered signal was above the threshold, it was considered as detected). When they adapted the algorithm to the step detection problem, [16] relied on the fact that the filtered signal showed great regularity: a small peak was always followed by a bigger one (respectively matching the foot lift and the heel strike). The time span of the second peak was defined as the peak-searching interval on the real acceleration signal. The maximum on that interval was considered a step.

Note that this step detection procedure only allows to detect steps but not to precisely know the start and end times of the step. Also, this method is not designed to perform properly during periods of no activity. We therefore added a post-processing step, which, once a step is detected, compares the standard deviation of a neighborhood around the detected peak to a noise level. The size of the neighborhood, as well as the noise level, are optimized by grid search so as to give the best performances.

In [16], the parameters used are f_{min} = 0 Hz, f_{max} = 20 Hz, N_{inte} = 0.1 s. The peak search procedure is performed sequentially: they select one peak every other peak, starting with the second one. With these parameters, we obtain of our database a recall of 99.53% and a precision of 51.20%. In fact, the peak-search procedure is not adapted and tend to detect several peaks within a step except of only one. This phenomenon has already been described by [17], [18].

In order to objectively compare our method to the Pan-Tomkins, we therefore tested several values for f_{min}, f_{max} and N_{inte}, as well as a more relevant peak-search procedure, which only selects the local maxima among the detected peaks, thus preventing multiple detections. In total, 5 parameters need to be optimized by grid search (filter bandpass $\times$ 2, integration window, neighborhood size and noise level). When optimized on the whole database so as to maximize the F-measure, the algorithm gives a 97.82% recall and a 95.72% precision. Detailed results are presented on Table III : while these scores are comparable with our method on healthy subjects, it is noticeable that Pan-Tomkins method has difficulty to deal with neurological and orthopedics subjects. In particular, on these subjects, an overdetection occurs, thus decreasing the precision. One possible explanation is that signals associated to pathological subjects tend to have smaller amplitudes and to be noisier that those belonging to healthy subjects. Thus, if the parameters of the filtering are inadapted, the preprocessing tends to increase the level of noise and to create artefacts that as misdetected as steps. This may be one limit of step detection methods based on signal processing: if the signals to be studied have different properties (noise, frequential content, amplitudes), it is tricky to find one unique processing adapted to all signals. This problem is overcome in template-based methods which inherently consider several models.

V. Discussion and perspectives

The main idea behind the algorithm is that there is not one typical step but rather several typical steps. This assumption is confirmed by the results obtained with state-of-the-art methods, which inherently define only one model and obtain degraded performances when confronted to pathological data. To go further, it is interesting to degrade the algorithm with only one template and look at the consequences on the results. A second grid search is conducted with the same parameters as in Section IV-A, but considering libraries composed of one unique template.

The best results are displayed on Table III. The metrics used in IV-C are also evaluated for this simulation and presented on Figure 3b. Surprisingly, the precision and recall are comparable with those obtained with five templates. The template used for detection in this method belongs to an orthopaedic subject with moderate troubles and lasts 82 samples (which is close to the median step duration on the database which is equal to 77 samples). It seems that the task of step counting can be performed with only one template. However, it can be seen on Figure 3b that using only one template creates a bias and a

systematic error on the estimation of end and duration times. Due to the large duration of the template used for detection, an overestimation of the duration often occurs.

We believe this simulation shows that the use of a single template is adapted for step counting on most subjects. The use of templates appears to give better performances than thresholding methods for step detection. However, if additional information are desired (such as the start and end times of the steps), it is crucial to take into account the variability of the subjects and of their locomotion, which can be done by adding several templates that reflect the different step durations and shapes.

Intuitively, the composition of the library is a fundamental feature of the algorithm. The choice of the templates to be used is an interesting question that can be answered in many different ways. In a medical context, templates can for example be introduced according to the characteristics and pathologies of the subjects to be studied: a neurologist may benefit from a library of templates composed of a selection of different neurological pathologies. They can also be specified by experts such as biomechanists who can extract typical steps covering the whole range of types of locomotion. Unsupervised machine learning techniques (such as dictionary learning) can also be used to automatically extract typical steps that are found on several exercises. It is also relevant to test semi-supervised techniques that could automatically choose the best library according to the input signal. All these leads are to be studied soon in collaboration with medical doctors and experts, and on more pathologies.

VI. Conclusion

We have described in this article a template-based method for step detection. This method, based on a greedy algorithm and a library of annotated step templates, achieves good and robust performances even with a small number of templates. When used on a large database composed of healthy and pathological subjects walking at different speeds, the method obtains a 98% recall and 98% precision. Moreover, the algorithm allows to detect the start and end times of each step with a very good precision even on pathological subjects.

Thanks to its robustness and low computational cost, this method could be extended to process signals acquired in free-living conditions. Indeed, the actual protocol is composed of a no activity period and a U-turn, and there is no obstacles for testing the algorithm on unconstrained walking. The algorithm may also be adapted to a lighter protocol using only waist accelerometer signals and based on the same principle.

Another topic of interest is the choice of the templates to be used in the library (as presented in Section V). Several selection processes could be implemented in order to automatically adapt to any type of pathology and to optimize the performances of the algorithm.

ACKNOWLEDGMENTS

The authors would like to thank N. Vayatis, D. Ricard, A. Yelnik, C. De Waele and T. Grégory for the thorough discussions, the design of the experiment, the data acquisition and clinical annotation. This work was supported by SATT Ile-de-France Innov.

References

[1] B. Mariani, "Assessment of foot signature using wearable sensors for clinical gait analysis and real-time activity recognition," Ph.D. dissertation, EPFL, 2012.

[2] M. Marschollek, M. Goevercin, K.-H. Wolf, B. Song, M. Gietzelt, R. Haux, and E. Steinhagen-Thiessen, "A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, Vancouver, Canada, 2008, pp. 1319–1322.

[3] A. Willemsen, F. Bloemhof, and H. Boom, "Automatic stance-swing phase detection from accelerometer data for peroneal nerve stimulation," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 12, pp. 1201–1208, 1990.

[4] B. Dijkstra, W. Zijlstra, E. Scherder, and Y. Kamsma, "Detection of walking periods and number of steps in older adults and patients with parkinson's disease: accuracy of a pedometer and an accelerometry-based method," *Age and ageing*, vol. 37, no. 4, pp. 436–441, 2008.

[5] J. Han, H. S. Jeon, B. S. Jeon, and K. S. Park, "Gait detection from three dimensional acceleration signals of ankles for the patients with parkinsons disease," in *Proceedings of the International Special Topic Conference on Information Technology in Biomedicine*, 2006.

[6] F. Ayachi, H. Nguyen, E. Goubault, P. Boissy, and C. Duval, "The use of empirical mode decomposition-based algorithm and inertial measurement units to auto-detect daily living activities of healthy adults," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, no. 99, 2016.

[7] R. Williamson and B. Andrews, "Gait event detection for fes using accelerometers and supervised machine learning," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 3, pp. 312–319, 2000.

[8] K. Tran, T. Le, and T. Dinh, "A high-accuracy step counting algorithm for iphones using accelerometer," in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2012, pp. 000 213–000 217.

[9] N. Naqvi, A. Kumar, A. Chauhan, and K. Sahni, "Step counting using smartphone-based accelerometer," *International Journal on Computer Science and Engineering (IJCSE)*, pp. 1–7, 2012.

[10] V. Renaudin, M. Susi, and G. Lachapelle, "Step length estimation using handheld inertial sensors," *Sensors*, vol. 12, no. 7, pp. 8507–8525, 2012.

[11] J. Kim, H. Jang, D.-H. Hwang, and C. Park, "A step, stride and heading determination for the pedestrian navigation system," *Journal of Global Positioning Systems*, vol. 3, no. 1-2, pp. 273–289, 2004.

[12] M. Oner, J. Pulcifer-Stump, P. Seeling, and T. Kaya, "Towards the run and walk activity classification through step detection-an android application," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2012, pp. 1980–1983.

[13] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 225–234.

[14] B. Auvinet, G. Berrut, C. Touzard, L. Moutel, N. Collet, D. Chaleil, and E. Barrey, "Reference data for normal subjects obtained with an accelerometric device," *Gait & posture*, vol. 16, no. 2, pp. 124–134, 2002.

[15] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 32, no. 3, pp. 230–236, 1985.

[16] H. Ying, C. Silex, A. Schnitzer, S. Leonhardt, and M. Schiek, "Automatic step detection in the accelerometer signal," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, Aachen, Germany, 2007, pp. 80–85.

[17] R. Libby, "A simple method for reliable footstep detection in embedded sensor platforms," Research report, 2012.

[18] G. Thüer and T. Verwimp, "Step detection algorithms for accelerometers," Master's thesis, Artesis University College of Antwerp, Belgium, 2008.

[19] M. Mladenov and M. Mock, "A step counter service for java-enabled devices using a built-in accelerometer," in *Proceedings of the International Workshop on Context-Aware Middleware and Services (COMSWARE 2009)*. ACM, 2009, pp. 1–5.

[20] E. Fortune, V. Lugade, M. Morrow, and K. Kaufman, "Step counts using a tri-axial accelerometer during activity," in *Proceedings of the American Society of Biomechanics Annual Meeting (ASB)*, Florida, USA, 2012.
[21] A. Salarian, H. Russmann, F. Vingerhoets, C. Dehollain, Y. Blanc, P. Burkhard, and K. Aminian, "Gait assessment in parkinson's disease: toward an ambulatory system for long-term monitoring," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 8, pp. 1434–1443, 2004.
[22] K. Ben Mansour, N. Rezzoug, and P. Gorce, "Comparison between several locations of gyroscope for gait events detection," *Computer methods in biomechanics and biomedical engineering*, pp. 1–2, 2015.

Bibliography

[1] Analysis of changepoint models.

[2] S. Adak. Time-dependent spectral analysis of nonstationary time series. *Journal of the American Statistical Association*, 93(444):1488–1501, 1998.

[3] D. Angelosante and G. B. Giannakis. Group lassoing change-points piece-constant AR processes. *EURASIP Journal on Advances in Signal Processing*, 70, 2012.

[4] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistical Surveys*, 4:40–79, 2010.

[5] S. Arlot, A. Celisse, and Z. Harchaoui. Kernel change-point detection. *arXiv preprint arXiv:1202.3878*, pages 1–26, 2012.

[6] J. Audiffren, R. Barrois-Müller, C. Provost, É. Chiarovano, L. Oudre, T. Moreau, C. Truong, A. Yelnik, N. Vayatis, P.-P. Vidal, C. De Waele, S. Buffat, and D. Ricard. Évaluation de l'équilibre et prédiction des risques de chutes en utilisant une Wii board balance. *Neurophysiologie Clinique/Clinical Neurophysiology*, 45(4-5):403, 2015.

[7] A. Aue and L. Horvàth. Structural breaks in time series. *Journal of Time Series Analysis*, 34:1–16, 2012.

[8] J. Bai. Least squares estimation of a shift in linear processes. *Journal of Time Series Analysis*, 15(5):453–472, 1994.

[9] J. Bai. Least absolute deviation of a shift. *Econometric Theory*, 11(3):403–436, 1995.

[10] J. Bai. Testing for parameter constancy in linear regressions: an empirical distribution function approach. *Econometrica*, 64(3):597–622, 1996.

[11] J. Bai. Estimating multiple breaks one at a time. *Econometric Theory*, 13(3):315–352, 1997.

[12] J. Bai. Estimation of a change-point in multiple regression models. *Review of Economic and Statistics*, 79(4):551–563, 1997.

[13] J. Bai. Estimation of multiple-regime regressions with least absolutes deviation. *Journal of Statistical Planning and Inference*, 74:103–134, 1998.

[14] J. Bai. Likelihood ratio tests for multiple structural changes. *Journal of Econometrics*, 91(2):299–323, 1999.

[15] J. Bai. Vector autoregressive models with structural changes in regression coefficients and in variance–covariance matrices. *Annals of Economics and Finance*, 1(2):301–336, 2000.

[16] J. Bai. Common breaks in means and variances for panel data. *Journal of Econometrics*, 157:78–92, 2010.

[17] J. Bai and P. Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, 66(1):47–78, 1998.

[18] J. Bai and P. Perron. Critical values for multiple structural change tests. *Econometrics Journal*, 6(1):72–78, 2003.

[19] J. Bai and P. Perron. Multiple structural change models: a simulation analysis. *Journal of Applied Econometrics*, 18:1–22, 2003.

[20] J. Bai and P. Perron. Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, 18(1):1–22, 2003.

[21] J. Bai, R. L. Lumsdaine, and J. H. Stock. Testing for and dating common breaks in multivariate time series. *The Review of Economic Studies*, 65(3):395–432, 1998.

[22] R. Baranowski and P. Fryzlewicz. wbs: wild binary segmentation for multiple change-point detection, 2015. URL https://cran.r-project.org/package=wbs.

[23] R. Baranowski, Y. Chen, and P. Fryzlewicz. not: narrowest-over-threshold change-point detection, 2016. URL https://cran.r-project.org/package=not.

[24] R. Barrois-Müller, L. Oudre, T. Moreau, C. Truong, N. Vayatis, S. Buffat, A. Yelnik, C. de Waele, T. Gregory, S. Laporte, P. P. Vidal, and D. Ricard. Quantify osteoarthritis gait at the doctor's office: a simple pelvis accelerometer based method independent from footwear and aging. *Computer Methods in Biomechanics and Biomedical Engineering*, 18 Suppl 1:1880–1881, 2015.

[25] 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.

[26] R. Barrois-Müller, D. Ricard, L. Oudre, L. Tlili, C. Provost, A. Vienne, P.-P. Vidal, S. Buffat, and A. Yelnik. Étude observationnelle du demi-tour à l'aide de capteurs inertiels chez les sujets victimes d'AVC et relation avec le risque de chute. *Neurophysiologie Clinique/Clinical Neurophysiology*, 46(4):244, 2016.

[27] D. Barry and J. A. Hartigan. Product partition models for change point problems. *The Annals of Statistics*, 20(1):260–279, 1992.

[28] D. Barry and J. A. Hartigan. A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 88(421):309–319, 1993.

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

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

[31] P. K. Bhattacharya. Some aspects of change-point analysis. In E. Carlstein, H.-G. Müller, and D. Siegmund, editors, *Change-point problems*. Institute of Mathematical Statistics, 23 edition, 1994.

[32] L. Birgé and P. Massart. Gaussian model selection. *Journal of the European Mathematical Society*, 3(3):203–268, 2001.

[33] L. Birgé and P. Massart. Minimal penalties for Gaussian model selection. *Probability Theory and Related Fields*, 138(1):33–73, 2007.

[34] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[35] L. Boysen, A. Kempe, V. Liebscher, A. Munk, and O. Wittich. Consistencies and rates of convergence of jump-penalized least squares estimators. *The Annals of Statistics*, 37(1):157–183, 2009.

[36] B. E. Brodsky and B. S. Darkhovsky. *Nonparametric methods in change point problems*. Springer Netherlands, 1993.

[37] B. E. Brodsky, B. S. Darkhovsky, A. Y. Kaplan, and S. L. Shishkin. A nonparametric method for the segmentation of the EEG. *Computer Methods and Programs in Biomedicine*, 60(2):93–106, 1999.

[38] T. Cai and L. Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory*, 57(7):4680–4688, 2011.

[39] A. Celisse, G. Marot, M. Pierre-Jean, and G. Rigaill. New efficient algorithms for multiple change-point detection with kernels. *ArXiv e-prints arXiv:1710.04556*, 2017.

[40] S. Chakar, É. Lebarbier, C. Levy-Leduc, and S. Robin. AR1seg: segmentation of an autoregressive Gaussian process of order 1, 2014. URL https://cran.r-project.org/package=AR1seg.

[41] S. Chakar, É. Lebarbier, C. Levy-Leduc, and S. Robin. A robust approach for estimating change-points in the mean of an AR(1) process. *Bernouilli Society for Mathematical Statistics and Probability*, 23(2):1408–1447, 2017.

[42] J. Chen and A. K. Gupta. Testing and locating variance changepoints with application to stock prices. *Journal of the American Statistical Association*, 92(438):739–747, 1997.

[43] J. Chen and A. K. Gupta. *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance*. 2011.

[44] Jie Chen and Arjun K. Gupta. *Parametric Statistical Change Point Analysis*. Birkhäuser Boston, 2011. doi: 10.1007/978-0-8176-4801-5.

[45] Jie Chen and Arjun K Gupta. *Parametric statistical change point analysis: With applications to genetics, medicine, and finance*. Springer Science & Business Media, 2011.

[46] S. S. Chen and P. S. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, page 8, Landsdowne, VA, 1998.

[47] H Chernoff and S Zacks. Estimating the Current Mean of a Normal Distribution which is Subjected to Changes in Time. *The Annals of Mathematical Statistics*, 35(3): 999–1018, 1964.

[48] S. Chib. Estimation and comparison of multiple change-point models. *Journal of Econometrics*, 86(2):221–241, 1998.

[49] H. Cho, M. Barigozzi, and P. Fryzlewicz. factorcpt: simultaneous change-point and factor analysis, 2016. URL https://cran.r-project.org/package=factorcpt.

[50] S. Clemencon, M. Depecker, and N. Vayatis. AUC optimization and the two-sample problem. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 360–368, Vancouver, Canada, 2009.

[51] A. Cleynen, G. Rigaill, and M. Koskas. Segmentor3IsBack: a fast segmentation algorithm, 2016. URL https://cran.r-project.org/package=Segmentor3IsBack.

[52] M. Csörgö and L. Horváth. *Limit theorems in change-point analysis*. Chichester, New York, 1997.

[53] M. A. Davenport and M. B. Wakin. Analysis of Orthogonal Matching Pursuit Using the Restricted Isometry Property. *IEEE Transactions on Information Theory*, 56(9): 4395–4401, 2010.

[54] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, 1997.

[55] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 209–216, Corvalis, Oregon, USA, 2007.

[56] E. I. Delatola, É. Lebarbier, T. Mary-Huard, F. Radvanyi, S. Robin, and J. Wong. SegCorr: a statistical procedure for the detection of genomic regions of correlated expression. *BMC Bioinformatics*, 18(1):1–15, 2017.

[57] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 53(8):2961–2974, 2005.

[58] B. M. Doyle and J. Faust. Breaks in the variability and comovement of G-7 economic growth. *The Review of Economics and Statistics*, 87(4):721–740, 2005.

[59] J. H. J. Einmahl and I. W. McKeague. Empirical likelihood based hypothesis testing. *Bernoulli*, 9(2):267–290, 2003.

[60] C. Erdman and J. W. Emerson. bcp: an R package for performing a Bayesian analysis of change point problems. *Journal of Statistical Software*, 23(3):1–13, 2007.

[61] R. Esteller, G. Vachtsevanos, J. Echauz, and B. Litt. A Comparison of waveform fractal dimension algorithms. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(2):177–183, 2001.

[62] P Fearnhead. Exact and efficient Bayesian inference for multiple changepoint problems. *Statistics and Computing*, 16(2):203–213, 2006.

[63] K. Frick, A. Munk, and H. Sieling. Multiscale change point inference. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 76(3):495–580, 2014.

[64] J. H. Friedman and L. C. Rafsky. Multivariate Generalizations of Wald-Wolfowitz and Smirnov two-sample tests. *The Annals of Statistics*, 7(4):697–717, 1979.

[65] P. Fryzlewicz. Unbalanced Haar technique for nonparametric function estimation. *Journal of the American Statistical Association*, 102(480):1318–1327, 2007.

[66] P. Fryzlewicz. breakfast: multiple change-point detection and segmentation, 2017. URL https://cran.r-project.org/package=breakfast.

[67] Piotr Fryzlewicz. Wild binary segmentation for multiple change-point detection. *Annals of Statistics*, 42(6):2243–2281, 2014.

[68] Y.-X. Fu and R. N. Curnow. Maximum likelihood estimation of multiple change points. *Biometrika*, 77(3):563–573, 1990.

[69] D. Garreau and S. Arlot. Consistent change-point detection with kernels. *arXiv preprint arXiv:1612.04740v3*, pages 1–41, 2017.

[70] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13:723–773, 2012.

[71] Y. Guédon. Exploring the latent segmentation space for the assessment of multiple change-point models. *Computational Statistics*, 28(6):2641–2678, 2013.

[72] 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.

[73] Z. Harchaoui and C. Lévy-Leduc. Multiple Change-Point Estimation With a Total Variation Penalty. *Journal of the American Statistical Association*, 105(492):1480–1493, 2010.

[74] Z. Harchaoui, F. Bach, and É. Moulines. Kernel change-point analysis. In *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, pages 609–616, Vancouver, Canada, 2008.

[75] 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.

[76] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*, volume 1. 2009.

[77] K. Haynes, I. A. Eckley, and P. Fearnhead. Computationally efficient changepoint detection for a range of penalties. *Journal of Computational and Graphical Statistics*, 26(1):134–143, 2017.

[78] K. Haynes, P. Fearnhead, and I. A. Eckley. A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing*, 27:1293–1305, 2017.

[79] H. He and T. S. Severini. Asymptotic properties of maximum likelihood estimators in models with multiple change points. *Bernoulli*, 16(3):759–779, 2010.

[80] G. Hébrail, B. Hugueney, Y. Lechevallier, and F. Rossi. Exploratory analysis of functional data via clustering and optimal segmentation. *Neurocomputing*, 73(7-9): 1125–1141, 2010.

[81] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 203–210, 2001.

[82] T. Hocking, G Rigaill, J.-P. Vert, and F. Bach. Learning sparse penalties for change-point detection using max margin interval regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 172–180, Atlanta, USA, 2013.

[83] T. Hocking, G. Schleiermacher, I. Janoueix-Lerosey, V. Boeva, J. Cappo, O. Delattre, F. Bach, and J.-P. Vert. Learning smoothing models of copy number profiles using breakpoint annotations. *BMC Bioinformatics*, 14(1):164, 2013.

[84] T. Hocking, G. Rigaill, and G. Bourque. PeakSeg: constrained optimal segmentation and supervised penalty learning for peak detection in count data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 324–332, Lille, France, 2015.

[85] K. Huang, R. Jin, Z. Xu, and C.-L. Liu. Robust metric learning by smooth optimization. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 244–251, Catalina Island, California, 2010.

[86] B. Hugueney, G. Hébrail, Y. Lechevallier, and F. Rossi. Simultaneous clustering and segmentation for functional data. In *Proceedings of 16th European Symposium on Artificial Neural Networks (ESANN)*, pages 281–286, Bruges, Belgium, 2009.

[87] 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.

[88] N. A. James and D. S. Matteson. ecp: an R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(7):1–25, 2014.

[89] N. A. James and D. S. Matteson. ecp: an R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(7), 2015.

[90] V. Jandhyala, S. Fotopoulos, I. Macneill, and P. Liu. Inference for single and multiple change-points in time series. *Journal of Time Series Analysis*, 34(4):423–446, 2013.

[91] K. Karagiannaki, A. Panousopoulou, and P. Tsakalides. An online feature selection architecture for Human Activity Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2522–2526, New Orleans, LA, USA, 2017.

[92] S. M. Kay and A. V. Oppenheim. *Fundamentals of Statistical Signal Processing, Volume II: Detection Theory*. Prentice Hall, 1993.

[93] M. G. Kendall. *Rank correlation methods*. Charles Griffin, London, England, 1970.

[94] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 289–296, 2001.

[95] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: a survey and novel approach. *Data Mining in Time Series Databases*, 57(1):1–22, 2004.

[96] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB) - Volume 30*, pages 180–191, Toronto, Canada, 2004.

[97] R. Killick and I. A. Eckley. changepoint: an R package for changepoint analysis. *Journal of Statistical Software*, 58(3):1–19, 2014.

[98] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500): 1590–1598, 2012.

[99] S. I. M. Ko, T. T. L. Chong, and P. Ghosh. Dirichlet process hidden Markov multiple change-point model. *Bayesian Analysis*, 10(2):275–296, 2015.

[100] K. Korkas and P. Fryzlewicz. wbsts: multiple change-point detection for nonstationary time series, 2015. URL https://cran.r-project.org/package=wbsts.

[101] K. Korkas and P. Fryzlewicz. Multiple change-point detection for non-stationary time series using wild binary segmentation. *Statistica Sinica*, 27(1):287–311, 2017.

[102] P. R. Krishnaiah. Review about estimation of change points. *Handbook of Statistics*, 7:375–402, 1988.

[103] B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research (JMLR)*, 19:341–376, 2009.

[104] W. R. Lai, M. D. Johnson, R. Kucherlapati, and P. J. Park. Comparative analysis of algorithms for identifying amplifications and deletions in array CGh data. *Bioinformatics*, 21(19):3763–3770, 2005.

[105] 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.

[106] M. Lavielle. Optimal segmentation of random processes. *IEEE Transactions on Signal Processing*, 46(5):1365–1373, 1998.

[107] M. Lavielle. Detection of multiples changes in a sequence of dependant variables. *Stochastic Processes and their Applications*, 83(1):79–102, 1999.

[108] M. Lavielle. Using penalized contrasts for the change-point problem. *Signal Processing*, 85(8):1501–1510, 2005.

[109] M. Lavielle and É. Moulines. Least-squares estimation of an unknown number of shifts in a time series. *Journal of Time Series Analysis*, 21(1):33–59, 2000.

[110] M. Lavielle and G. Teyssière. Detection of multiple change-points in multivariate time series. *Lithuanian Mathematical Journal*, 46(3), 2006.

[111] M. Lavielle and G. Teyssière. Adaptive detection of multiple change-points in asset price volatility. In *Long-Memory in Economics*, pages 129–156. Springer Verlag, Berlin, Germany, 2007.

[112] É. Lebarbier. Detecting multiple change-points in the mean of gaussian process by model selection. *Signal Processing*, 85(4):717–736, 2005.

[113] E. L. Lehman and J. P. Romano. *Testing Statistical Hypotheses*, volume 101. springer, 3 edition, 2006. ISBN 0387988645. doi: 10.1198/jasa.2006.s100.

[114] C. Lévy-Leduc and F. Roueff. Detection and localization of change-points in high-dimensional network traffic data. *The Annals of Applied Statistics*, 3(2):637–662, 2009.

[115] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[116] G. Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

[117] A. Lung-Yut-Fong, C. Lévy-Leduc, and O. Cappé. Distributed detection/localization of change-points in high-dimensional network traffic data. *Statistics and Computing*, 22(2):485–496, 2012.

[118] A. Lung-Yut-Fong, C. Lévy-Leduc, and O. Cappé. Homogeneity and change-point detection tests for multivariate data using rank statistics. *Journal de la Société Française de Statistique*, 156(4):133–162, 2015.

[119] P. C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

[120] R. Maidstone. Efficient Analysis of Complex Changepoint Models. page 34, 2013.

[121] R. Maidstone, T. Hocking, G. Rigaill, and P. Fearnhead. On optimal multiple changepoint algorithms for large data. *Statistics and Computing*, 27(2):519–533, 2017.

[122] C. L. Mallows. Some comments on Cp. *Technometrics*, 15(4):661–675, 1973.

[123] A. F. Martínez and R. H. Mena. On a Nonparametric Change Point Detection Model in Markovian Regimes. *Bayesian Analysis*, 9(4):823–858, 2014.

[124] C. F. H. Nam, J. A. D. Aston, and A. M. Johansen. Quantifying the uncertainty in change points. *Journal of Time Series Analysis*, 33:807–823, 2012.

[125] G. P. Nason, R. von Sachs, and G. Kroisandt. Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(2):271–292, 2000.

[126] A. B. Olshen, E. S. Venkatraman, R. Lucito, and M. Wigler. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, 5(4): 557–572, 2004.

[127] L. Oudre, A. Lung-Yut-Fong, and P. Bianchi. Segmentation of accelerometer signals recorded during continuous treadmill walking. In *Proceedings of the 19th European Signal Processing Conference (EUSIPCO)*, pages 1564–1568, 2011.

[128] L. Oudre, R. Barrois-Müller, T. Moreau, C. Truong, R. Dadashi, T. Grégory, D. Ricard, N. Vayatis, C. De Waele, A. Yelnik, and P.-P. Vidal. Détection automatique des pas à partir de capteurs inertiels pour la quantification de la marche en consultation. *Neurophysiologie Clinique/Clinical Neurophysiology*, 45(4-5):394, 2015.

[129] L. Oudre, T. Moreau, and C. Truong. Détection de pas à partir de données d'accélérométrie. In *Proceedings of the Groupe de Recherche et d'Etudes en Traitement du Signal et des Images (GRETSI)*, Lyon, France, 2015.

[130] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–105, 1954.

[131] E. S. Page. A test for a change in a parameter occurring at an unknown point. *Biometrika*, 42:523–527, 1955.

[132] P. Perron. Dealing with structural breaks. *Palgrave handbook of econometrics*, 1(2): 278–352, 2006.

[133] P. Perron and Z. Qu. Estimating restricted structural change models. *Journal of Econometrics*, 134(2):373–399, 2006.

[134] F. Picard, S. Robin, M. Lavielle, C. Vaisse, and J.-J. Daudin. A statistical approach for array CGH data analysis. *BMC Bioinformatics*, 6(1):27, 2005.

[135] T. Pohlert. trend: non-parametric trend tests and change-point detection, 2017. URL https://cran.r-project.org/package=trend.

[136] R. Prescott Adams and D. J. C. MacKay. Bayesian Online Changepoint Detection. Technical report, 2007.

[137] Z. Qu and P. Perron. Estimating and testing structural changes in multivariate regressions. *Econometrica*, 75(2):459–502, 2007.

[138] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[139] G. Rigaill. A pruned dynamic programming algorithm to recover the best segmentations with 1 to K_max change-points. *Journal de la Société Française de Statistique*, 156(4):180–205, 2015.

[140] G. Ross. Parametric and nonparametric sequential change detection in R: the cpm package. *Journal of Statistical Software*, 66(3):1–20, 2015.

[141] G. J. Ross. Parametric and nonparametric sequential change detection in R: the cpm package. *Journal of Statistical Software*, 66(3), 2015.

[142] B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, Cambridge, USA, 2002.

[143] M. Schultz and J. Thorsten. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems 16 (NIPS)*, Vancouver, Canada, 2003.

[144] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

[145] N. Seichepine, S. Essid, C. Fevotte, and O. Cappé. Piecewise constant nonnegative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6721–6725, Florence, Italy, 2014.

[146] A. Sen and M. S. Srivastava. On tests for detecting change in mean. *The Annals of Statistics*, 3(1):98–108, 1975.

[147] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge university press, 2004.

[148] J. J. Shen and N. R. Zhang. SeqCBS: CN profiling using sequencing and CBS, 2012.

[149] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 776–790, 2002.

[150] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf. Injective Hilbert space embeddings of probability measures. In *Proceedings of the 21st Conference on Learning Theory (COLT)*, pages 9–12, Helsinki, Finland, 2008.

[151] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[152] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[153] C. Truong, L. Oudre, and N. Vayatis. Segmentation de signaux physiologiques par optimisation globale. In *Proceedings of the Groupe de Recherche et d'Etudes en Traitement du Signal et des Images (GRETSI)*, Lyon, France, 2015.

[154] C. Truong, L. Oudre, and N. Vayatis. Penalty learning for changepoint detection. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017.

[155] C. Truong, L. Oudre, and N. Vayatis. ruptures, change point detection in Python, 2018. URL http://mloss.org/software/view/700/.

[156] C. Truong, L. Oudre, and N. Vayatis. A review of change point detection. *arXiv preprint arXiv:1801.00718*, pages 1–31, 2018.

[157] C. Truong, L. Oudre, and N. Vayatis. ruptures: change point detection in Python. *ArXiv e-prints arXiv:1801.00826*, pages 1–5, 2018.

[158] C. Truong, L. Oudre, and N. Vayatis. Greedy kernel change point detection with an application to physiological signals. *Submitted*, pages 1–5, 2018.

[159] E. S. Venkatraman and A. B. Olshen. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, 23(6):657–663, 2007.

[160] J.-P. Vert and K. Bleakley. Fast detection of multiple change-points shared by many signals using group LARS. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, volume 1, pages 2343–2351, Vancouver, Canada, 2010.

[161] L. Y. Vostrikova. Detecting disorder in multidimensional random processes. *Soviet Math. Dokl.*, 24:55–59, 1981.

[162] H. Vullings, M. Verhaegen, and H. Verbruggen. ECG segmentation using time-warping. In *Lecture notes in computer science*, pages 275–286. Springer, 1997.

[163] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

[164] H. Willenbrock and J. Fridlyand. A comparison study: applying segmentation to array CGH data for downstream analyses. *Bioinformatics*, 21(22):4084–4091, 2005.

[165] E. P. Xing, M. I. Jordan, and S. J. Russell. Distance metric learning, with application to clustering with side-Information. In *Advances in Neural Information Processing Systems 21 (NIPS 2003)*, pages 521–528, 2003.

[166] Y.-C. Yao. Estimating the number of change-points via Schwarz' criterion. *Statistics and Probability Letters*, 6(3):181–189, 1988.

[167] Y.-C. Yao and S. T. Au. Least-squares estimation of a step function. *Sankhyā: The Indian Journal of Statistics, Series A*, 51(3):370–381, 1989.

[168] A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. strucchange: an R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2): 1–38, 2002.

[169] J. Zhang. Powerful two-sample tests based on the likelihood ratio. *Technometrics*, 48 (1):95–103, 2006.

[170] N. R. Zhang and D. O. Siegmund. A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1):22–32, 2007.

[171] C. Zou, G. Yin, F. Long, and Z. Wang. Nonparametric maximum likelihood approach to multiple change-point problems. *The Annals of Statistics*, 42(3):970–1002, 2014.

université PARIS-SACLAY | École doctorale de mathématiques Hadamard (EDMH)

Titre : Détection de ruptures multiples – application aux signaux physiologiques

Mots clés : détection de ruptures, traitement du signal, méthodes à noyaux

Résumé : Ce travail traite de la détection de ruptures multiples dans des signaux physiologiques multivariés. L'objectif est de fournir des algorithmes de détection (i) capables de gérer de longues séries, (ii) utilisables sur une large gamme de problèmes réels et (iii) capables d'incorporer la connaissance d'experts médicaux.
La première contribution de cette thèse est un algorithme sous-optimal de détection, qui peut s'adapter à des contraintes de complexité, tout en conservant la robustesse des méthodes optimales. Dans le contexte des sauts de moyenne, un résultat de consistance asymptotique est prouvé. Cette stratégie gloutonne est étendue à d'autres types de ruptures, grâce aux espaces de Hilbert à noyaux reproduisant. Des expériences sur des signaux réels montrent que ces approches sont plus précises que les approches sous-optimales standards et plus rapides que les méthodes optimales.
La deuxième contribution de cette thèse consiste en deux algorithmes supervisés de calibration automatique. Ils se reposent tous les deux sur des signaux annotés par des experts. La première approche apprend le paramètre de lissage pour la détection pénalisée d'un nombre inconnu de ruptures. La seconde procédure apprend une transformation non-paramétrique de l'espace de représentation du signal. Les résultats expérimentaux montrent que ces méthodes supervisées ont de meilleures performances que les méthodes non-supervisées, particulièrement dans le cas des signaux physiologiques, où la notion de rupture dépend du phénomène physiologique d'intérêt.
Toutes les contributions algorithmiques de cette thèse sont disponibles dans **ruptures**, un logiciel libre, en Python et entièrement documenté.

Title : Multiple change point detection. Application to physiological signals

Keywords : change point detection, signal processing, kernel methods

Abstract : This work addresses the problem of detecting multiple change points in multivariate physiological signals. The objective of this thesis is to provide change point detection algorithms that (i) can handle long signals, (ii) can be applied on a wide range of real-world scenarios, and (iii) can incorporate the knowledge of medical experts.
The first contribution of this thesis is a sub-optimal change detection algorithm that can accommodate time complexity constraints while retaining most of the robustness of optimal procedures. In the context of mean-shifts, asymptotic consistency of estimated change points is proved. This greedy strategy is extended to other types of changes, by using reproducing kernel Hilbert spaces. Experiments on real-world signals show that those approaches are more accurate than standard sub-optimal algorithms and faster than optimal algorithms.
The second contribution of this thesis consists in two supervised algorithms for automatic calibration. Both rely on labeled examples, which in our context, consist in segmented signals. The first approach learns the smoothing parameter for the penalized detection of an unknown number of changes. The second procedure learns a non-parametric transformation of the representation space. Results show that those supervised algorithms outperform unsupervised algorithms, especially in the case of physiological signals, where the notion of change heavily depends on the physiological phenomenon of interest.
All algorithmic contributions of this thesis can be found in **ruptures**, a thoroughly documented open-source Python library.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France