New efficient algorithms for multiple change-point detection with kernels

Alain Celisse, Guillemette Marot, Morgane Pierre-Jean, Guillem Rigaill

New efficient algorithms for multiple change-point detection with kernels

A. Célisse[c,e], G. Marot[a,c], M. Pierre-Jean[a,b], G.J. Rigaill[b,d]

[a] ***Univ. Lille Droit et Santé*** *EA 2694 - CERIM, F-59000 Lille, France*
[b] ***UMR 8071 CNRS - Université d'Evry - INRA*** *Laboratoire Statistique et Génome Evry*
[c] ***Inria Lille Nord Europe*** *Équipe-projet Inria* MODAL
[d] ***Institute of Plant Sciences Paris-Saclay****, UMR 9213/UMR1403, CNRS, INRA, Université Paris-Sud, Université d'Evry, Université Paris-Diderot, Sorbonne Paris-Cité*
[e] ***Univ. Lille Sciences et Technologies, CNRS,*** *UMR 8524 - Laboratoire Paul Painlevé, F-59000 Lille, France*

Abstract

Several statistical approaches based on reproducing kernels have been proposed to detect abrupt changes arising in the full distribution of the observations and not only in the mean or variance. Some of these approaches enjoy good statistical properties (oracle inequality, ...). Nonetheless, they have a high computational cost both in terms of time and memory. This makes their application difficult even for small and medium sample sizes ($n < 10^4$). This computational issue is addressed by first describing a new efficient and exact algorithm for kernel multiple change-point detection with an improved worst-case complexity that is quadratic in time and linear in space. It allows dealing with medium size signals (up to $n \approx 10^5$). Second, a faster but approximation algorithm is described. It is based on a low-rank approximation to the Gram matrix. It is linear in time and space. This approximation algorithm can be applied to large-scale signals ($n \geq 10^6$). These exact and approximation algorithms have been implemented in `R` and `C` for various kernels. The computational and statistical performances of these new algorithms have been assessed through empirical experiments. The runtime of the new algorithms is observed to be faster than that of other considered procedures.

Finally, simulations confirmed the higher statistical accuracy of kernel-based approaches to detect changes that are not only in the mean. These simulations also illustrate the flexibility of kernel-based approaches to analyze complex biological profiles made of DNA copy number and allele B frequencies.

An R package implementing the approach will be made available on github.

Keywords: Kernel method, Gram matrix, nonparametric change-point detection, model selection, algorithms, dynamic programming, DNA copy number, allele B fraction

1. Introduction

In this paper we consider the multiple change-point detection problem [13] where the goal is to recover abrupt changes arising in the distribution of a sequence of n independent random variables $X_1, \ldots, X_n$ observed at respective time $t_1 < t_2 < \ldots < t_n$.

State-of-the-art. Many parametric models (Normal, Poisson,...) have been proposed [29, 47, 17]. These models allow detecting different types of changes: in the mean, in the variance and in both the mean and variance (see also [29, 33, 44]). Efficient algorithms and heuristics have been proposed for these models. Some of them scale in $\mathcal{O}(n\log(n))$ or even in $\mathcal{O}(n)$. In practice, these parametric approaches have proven to be successful for various application fields (see for example [31, 15]). However one of their main drawbacks is their lack of flexibility. For instance, any change of distributional assumption requires the development of a new dedicated inference scheme.

By contrast, the recently proposed kernel change-point detection approach [28, 3] is more generic. It has the potential to detect any change arising in the distribution, which is not easily captured by standard parametric models. More precisely in this approach, the observations are first mapped into a Reproducing Kernel Hilbert Space (RKHS) through a kernel function [4]. The difficult problem of detecting changes in the distribution is then recast as simply detecting changes in the mean element of observations in the RKHS, which is made possible using the well-known kernel trick.

One practical limitation of this kernel-based approach is its considerable computational cost owing to the use of a $n \times n$ Gram matrix combined with a dynamic programming algorithm [6]. More precisely [28] described a dynamic programming algorithm to recover the best segmentation from 1 to $D_{\max}$ segments. They claim that their algorithm has a $\mathcal{O}(D_{\max}n^2)$ time complexity. However, the latter is not described in full details and its straightforward implementation is not efficient. First, it requires the storage of a $n \times n$ cost matrix (personal communication with the first author of [28] who was kind enough to send us his code). Thus the algorithm has a $\mathcal{O}(n^2)$ space complexity, which is a severe limitation with nowadays sample sizes. For instance analyzing a signal of length $n = 10^5$ requires storing a $10^5 \times 10^5$ matrix of doubles, which takes 80 GB. Second, computing the cost matrix is not straightforward. In fact simply using formula (8) of [28] to compute each term of this cost matrix leads to an $\mathcal{O}(n^4)$ time complexity.

Contributions. The present paper contains several contributions to the computational aspects and the statistical performance of the kernel change-point procedure introduced by [3].

The first one is to describe a new algorithm to simultaneously perform the dynamic programming step of [28] and also compute the required elements of the cost matrix on the fly. On the one hand, this algorithm has a complexity of order $\mathcal{O}(D_{\max}n^2)$ in time and $\mathcal{O}(D_{\max}n)$ in space (including both the dynamic programming and the cost matrix computation). We also emphasize that this improved space complexity comes without an increased time complexity. This is a great algorithmic improvement upon the change-point detection approach described by [3] since it allows the efficient analysis of signals with up to $n = 10^5$ data-points in a matter of a few minutes on a standard laptop.

On the other hand, our approach is generic in the sense that it works for any positive semidefinite kernels. Importantly one cannot expect to exactly recover the best segmentations from 1 to $D_{\max}$ segments in less than $\mathcal{O}(D_{\max}n^2)$ without additional specific assumptions on the kernel. Indeed,

computing the cost of a given segmentation has already a time complexity of order $\mathcal{O}(n^2)$.

It is also noticeable that our algorithm can be applied to other existing strategies such as the so-called ECP [40]. To be specific, we show that the *divisive clustering algorithm* it is based on and that provides an approximate solution with a complexity of order $O(n^2)$ in time and space can be replaced by our algorithm that provides the exact solution with the same time complexity and reduced memory complexity.

Our second contribution is a new algorithm dealing with larger signals ($n > 10^5$) based on a low-rank approximation to the Gram matrix. This computational improvement is possible at the price of an approximation. It returns approximate best segmentations from 1 to $D_{\max}$ segments with a complexity of order $\mathcal{O}(D_{\max}p^2n)$ in time and $\mathcal{O}((D_{\max}+p)n)$ in space, where p is the rank of the approximation.

The last contribution of the paper is the empirical assessment of the statistical performance of the KCP procedure introduced by [3]. This empirical analysis is carried out in the biological context of detecting abrupt changes from a two-dimensional signal made of DNA copy numbers and allele B fractions [35]. The assessment is done by comparing our approach to state-of-the-art alternatives on resampled real DNA copy number data [45, 40]. This illustrates the versatility of the kernel-based approach. To be specific this approach allows the detection of changes in the distribution of such complex signals without explicitly modeling the type of change we are looking for.

The remainder of the paper is organized as follows. In Section 2, we describe our kernel-based framework and detail the connection between detecting abrupt changes in the distribution and model selection as described in [3]. A slight generalization of the KCP procedure [3] is also derived in Section 2.5 by introducing a new parameter ℓ encoding an additional constraint on the minimal length of any candidate segment. This turns out to be particularly useful in low signal-to-noise ratio settings. The versatility of this kernel-based framework is emphasized in Section 2.6 where it is shown how the ECP approach [40] can be rephrased in terms of kernels. Our main algorithmic improvements are detailed and justified in Section 3. We empirically illustrate the improved runtime of our algorithm and compare it to the ones of ECP and RBS in Section 3.1.3. In Section 3.2 we detail our faster (but approximate) algorithm used to analyze larger profiles ($n > 10^5$). It is based on the combination of a low-rank approximation to the Gram matrix and the binary segmentation heuristic [53]. An empirical comparison of the runtimes of the exact and approximate algorithms is provided in Section 3.2.3. Finally, Section 4 illustrates the statistical performance of our kernel-based change-point procedure in comparison with state-of-the-art alternatives in the context of biological signals such as DNA copy numbers and allele B fractions [35].

2. Kernel framework

In this section we recall the framework of [28] where detecting changes in the distribution of a complex signal is rephrased as detecting changes of the mean element of a sequence of points in a

Hilbert space. Then we detail the so-called KCP (Kernel Change-point Procedure) [3], which has been proved to be optimal in terms of an oracle inequality.

2.1. Notation

Let $X_1, X_2, \ldots, X_n \in \mathcal{X}$ be a time-series of n independent random variables, where $\mathcal{X}$ denotes any set assumed to be *separable* [19] throughout the paper. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denote a symmetric positive semi-definite kernel [4], and $\mathcal{H}$ be the associated reproducing kernel Hilbert space (RKHS). We refer to [11] for an extensive presentation about kernels and RKHS. Let us also introduce the canonical feature map $\Phi : \mathcal{X} \to \mathcal{H}$ defined by $\Phi(x) = k(x, \cdot) \in \mathcal{H}$, for every $x \in \mathcal{X}$. This canonical feature map allows to define the inner product on $\mathcal{H}$ from the kernel k, by

$$\forall x, y \in \mathcal{X}, \qquad \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}} = k(x, y). \tag{1}$$

The asset of kernels. One main advantage of kernels is to enable dealing with complex data of any type provided a kernel can be defined. In particular no vector space structure is required on $\mathcal{X}$. For instance $\mathcal{X}$ can be a set of DNA sequences, a set of graphs or a set of distributions to name but a few examples (see [24] for various instances of $\mathcal{X}$ and related kernels). Therefore, as long as a kernel k can be defined on $\mathcal{X}$, any element $x \in \mathcal{X}$ is mapped, through the canonical feature map Φ, to an element of the Hilbert space $\mathcal{H}$. This provides a unified way to deal with different types of (simple or complex) data. Then for every index $1 \leq t \leq n$, let us note

$$Y_t = \Phi(X_t) \in \mathcal{H}. \tag{2}$$

From now on, we will only consider the following sequence $Y_1, \ldots, Y_n \in \mathcal{H}$ of independent Hilbert-valued random vectors.

The kernel trick. As a space of functions from $\mathcal{X}$ to $\mathbb{R}$, the RKHS $\mathcal{H}$ can be infinite dimensional. From a computational perspective one could be worried that manipulating such objects is computationally prohibitive. However this is not the case and our algorithm relies on the so-called *kernel trick*, which consists in translating any inner product in $\mathcal{H}$ in terms of the kernel k by use of Eq. (1). For every $1 \leq i, j \leq n$, it results

$$\langle Y_i, Y_j \rangle_{\mathcal{H}} = k(X_i, X_j) = \mathbf{K}_{i,j},$$

where $\mathbf{K}_{i,j}$ denotes the (i, j)-th coefficient of the $n \times n$ Gram matrix $\mathbf{K} = [\, k(X_i, X_j) \,]_{1 \leq i,j \leq n}$.

2.2. Detecting changes in the distribution using kernels

Let us consider the model introduced by [3], which connects every Y_t to its "mean" $\mu_t^\star \in \mathcal{H}$ by

$$\forall\ 1 \leq t \leq n, \qquad Y_t = \Phi(X_t) = \mu_t^\star + \epsilon_t \in \mathcal{H}, \tag{3}$$

where $\mu_t^\star$ denotes the *mean element* associated with the distribution $\mathbb{P}_{X_t}$ of X_t, and $\epsilon_t = Y_t - \mu_t^\star$. Let us also recall [38] that if $\mathcal{X}$ is separable and $\mathbb{E}\,[\, k(X_t, X_t) \,] < +\infty$, then $\mu_t^\star$ exists and is defined as the unique element in $\mathcal{H}$ such that

$$\forall\ f \in \mathcal{H}, \quad \langle \mu_t^\star, f \rangle_{\mathcal{H}} = \mathbb{E}\, \langle \Phi(X_t), f \rangle_{\mathcal{H}}\,. \tag{4}$$

For characteristic kernels [50], a change in the distribution of X_t implies a change in the mean element $\mu_t^\star$, that is

$$\forall\ 1 \leq i \neq j \leq n, \qquad \mathbb{P}_{X_i} \neq \mathbb{P}_{X_j} \Rightarrow \mu_i^\star \neq \mu_j^\star, \tag{5}$$

the converse implication being true by definition of $\mu_t^\star$ in Eq. (4). The idea behind kernel change-point detection [3] is to translate the problem of detecting changes in the distribution into detecting changes in the mean of Hilbert-valued vectors.

Remark 1. *When considering $\mathcal{X} \subset \mathbb{R}^q$ for some integer $q > 0$, several classical kernels are characteristic. For instance,*

- *The Gaussian kernel: $k(x,y) = e^{-\|x-y\|^2/\delta}$, with $x, y \in \mathbb{R}^q$ and $\delta > 0$,*
- *The Laplace kernel: $k(x,y) = e^{-\|x-y\|/\delta}$, with $x, y \in \mathbb{R}^q$ and $\delta > 0$,*
- *The exponential kernel: $k(x,y) = e^{-\langle x,y\rangle_q/\delta}$, with $x, y \in \mathbb{R}^q$ and $\delta > 0$,*

where $\|\cdot\|$ and $\langle\cdot,\cdot\rangle_q$ respectively denote the usual Euclidean norm and inner product in $\mathbb{R}^q$. The energy-based kernel discussed in Section 2.6 is also a characteristic kernel (see Lemma 1 in [41]). However, with more general sets $\mathcal{X}$, building a characteristic kernel is challenging as illustrated by [50] and [14].

Let us also notice that the procedure developed by [3] can be seen as a "kernelized version" of the procedure proposed by [37], which was originally designed to detect changes in the mean of real-valued variables.

2.3. Statistical framework

From Eq. (5) it results that any sequence of abrupt changes in the distribution along the time corresponds to a sequence of D^* true change-points $1 = \tau_1^* < \tau_2^* < \ldots < \tau_{D^*}^* \leq n$ (with $\tau_{D^*+1}^* = n+1$ by convention) such that

$$\mu_1^* = \ldots = \mu_{\tau_1^*-1}^* \neq \mu_{\tau_1^*}^* = \ldots = \mu_{\tau_2^*-1}^* \neq \ldots \neq \mu_{\tau_{D^*}^*}^* = \ldots = \mu_n^*.$$

In other words we get that $\mu^* = (\mu_1^*, \ldots, \mu_n^*)' \in \mathcal{H}^n$ is piecewise constant.

From a set of D candidate change-points $1 = \tau_1 < \ldots < \tau_D \leq n$, let τ be defined by

$$\tau = (\tau_1, \tau_2, \ldots, \tau_{D-1}, \tau_D)\,,$$

with the convention $\tau_1 = 1$ and $\tau_{D+1} = n+1$. With a slight abuse of notation, we also call τ the segmentation of $\{1, \ldots, n\}$ associated with the change-points $1 = \tau_1 < \ldots < \tau_D \leq n$. The estimator $\hat{\mu}^\tau = (\hat{\mu}_1^\tau, \ldots, \hat{\mu}_n^\tau)' \in \mathcal{H}^n$ of $\mu^* = (\mu_1^*, \ldots, \mu_n^*)'$ proposed by [3] is defined by

$$\forall\ 1 \leq i \leq D, \quad \forall\ t \in \{\tau_i, \ldots, \tau_{i+1} - 1\}\,, \quad \hat{\mu}_t^\tau = \frac{1}{\tau_{i+1} - \tau_i} \sum_{t'=\tau_i}^{\tau_{i+1}-1} Y_{t'}.$$

The performance of $\hat{\mu}^\tau$ is measured by the quadratic risk:

$$\mathcal{R}(\hat{\mu}^\tau) = \mathbb{E}\left[\, \|\mu^* - \hat{\mu}^\tau\|_{\mathcal{H},n}^2 \,\right] = \mathbb{E}\left[\sum_{i=1}^n \|\mu_i^* - \hat{\mu}_i^\tau\|_{\mathcal{H}}^2 \right],$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm in the Hilbert space $\mathcal{H}$.

2.4. Model selection

If the signal-to-noise ratio is small, [3] emphasized that all true change-points cannot be recovered without including false change-points. This leads them to define the best segmentation τ^* (for a finite sample size) as

$$\tau^* = \arg\min_{\tau \in \mathcal{T}_n} \|\mu^* - \hat{\mu}^\tau\|_{\mathcal{H},n},$$

where $\mathcal{T}_n$ denotes the collection of all possible segmentations τ of $\{1, \ldots, n\}$ with at most $D_{\max}$ segments. When the signal-to-noise ratio is large enough, τ^* coincides with the true segmentation.

As a surrogate to the previous unknown criterion, [3] optimize the following penalized criterion

$$\hat{\tau} = \arg\min_{\tau \in \mathcal{T}_n} \left\{ \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} + \text{pen}(\tau) \right\}, \qquad \text{with} \quad \text{pen}(\tau) = c_1 D_\tau + c_2 \log \binom{n-1}{D-1}, \tag{6}$$

where $Y = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$, $c_1, c_2 > 0$ are constants to be fixed, and D_τ denotes the number of segments of the segmentation τ. Since this penalty only depends on τ through D_τ, optimizing (6) can be formulated as a two-step procedure. The first step consists in solving:

$$\forall\ 1 \leq D \leq D_{\max}, \quad \hat{\tau}_D = \arg\min_{\tau \in \mathcal{T}_D} \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n}, \tag{7}$$

where $\mathcal{T}_D$ denotes the set of segmentations with D segments. This optimization problem, which is usually solved by dynamic programming [6, 47], is computationally hard since the cardinality of $\mathcal{T}_D$ is $\binom{n-1}{D-1}$. The second step is to straightforwardly optimize:

$$\hat{D} = \arg\min_{1 \leq D \leq D_{\max}} \left\{ \|Y - \hat{\mu}_{\hat{\tau}_D}\|^2_{\mathcal{H},n} + \text{pen}(\hat{\tau}_D) \right\} \qquad \text{and} \quad \hat{\tau} = \hat{\tau}_{\hat{D}}. \tag{8}$$

The right-most term in the penalty (6) accounts for the number of candidate segmentations with D segments (see the comments of Theorem 2 in [3]). Intuitively this term balances the trend of the estimator (7) to overfit because of the large number of candidate segmentations.

Remark 2. *It is important to notice that the above two-step procedure depends on the hyper-parameter $D_{\max}$, which is the maximum number of segments of the candidate segmentations.*

In fact choosing an appropriate $D_{\max}$ is related to the calibration of constants c_1 and c_2 in the penalty term of (6)*. Since the optimal values of c_1 and c_2 depend (at least) on the variance of the signal at hand, they have to be calibrated in a data-driven way. Here they have been calibrated by using the so-called* slope heuristic *technique described in [3] (see also the numerical experiments in Section 4 for more details). In particular $D_{\max}$ has to be chosen large enough to make the slope heuristic work well. Given some prior knowledge of an adequate range of values for D taking D_{max} to be 10 to 20 times larger than that seems to work well in practice. Typically for copy number data (see Section 4.1.1) one rarely expect more than 10 change-points per chromosome and taking $D_{max} \approx 100$ or 200 often make sense.*

From a theoretical point of view, this model selection procedure has been proved to be optimal in terms of an oracle inequality by [3]. This is the usual non-asymptotic optimality result for model

selection procedures [12]. This procedure has also been proved to provide consistent estimates of the change-points [23]. However, from a computational point of view, the first step (*i.e.* solving Eq. (7)) remains challenging. Indeed existing dynamic programming algorithms are time and space consuming when used in the kernel framework as it will be clarified in Section 3.1.1. The main purpose of the present paper is to provide a new computationally efficient algorithm to solve Eq. (7). Our new algorithm has a reduced space and time complexity and allows the analysis of signals larger than $n = 10^4$.

2.5. Low signal-to-noise and minimal length of a segment

In settings where the signal-to-noise ratio is weak (see for instance Figure 4 where the tumor percentage is low) change-point detection procedures are more likely to put changes in noisy regions. This results in overfitting and meaningless small segments [1]. A common solution is to include a constraint on the minimum length ℓ of segments. For instance by default ECP enforces that the estimated segmentation has segments with at least $\ell = 30$ points [32].

One important side effect of this constraint on ℓ is that the total number of candidate segmentations with D segments quickly decreases with ℓ. Therefore the penalty in (6) has to be modified.

The following lemma gives the cardinality of this set of segmentations.

Lemme 1. *Let $\mathcal{T}_n^\ell(D)$ denote the set of segmentations of $(1, \dots, n)$ in exactly $D \geq 1$ segments such that the length of each segment is at least $\ell \geq 1$. Then the cardinality of $\mathcal{T}_n^\ell(D)$ satisfies*

$$\text{Card}\left(\mathcal{T}_n^\ell(D)\right) = \binom{n - D(\ell - 1) - 1}{D - 1}.$$

Let us notice that if $\ell = 1$, one recovers the usual cardinality that is used in the penalty (see Eq. (6)). As an illustration of the influence of the constraint on ℓ, let us consider the set-up where $n = 100$, $D = 10$, and $\ell = 10$. Then the size of the unconstrained set of segmentations with 10 segments $\mathcal{T}_{100}(10) = \mathcal{T}_{100}^1(10)$ is $\text{Card}\left(\mathcal{T}_{100}(10)\right) \approx 1.7 \cdot 10^{12}$, whereas the constrained set $\mathcal{T}_{100}^{10}(10)$ is smaller since its cardinality is equal to 1.

Proof of Lemma 1. The proof consists in showing that there is a one-to-one mapping between the set $\mathcal{T}_n^\ell(D)$ of segmentations of $(1, \dots, n)$ with D segments of length at least $\ell \geq 1$, and the set $\mathcal{S}_n^\ell(D)$ of segmentations of $(1, \dots, n - D(\ell - 1))$ with D (non-empty) segments.

Let us consider one segmentation $\tau \in \mathcal{T}_n^\ell(D)$. Since each segment of τ is of length at least ℓ, let us remove $\ell - 1$ points from the left edge of each of the D segments. Then the resulting segmentation belongs to $\mathcal{S}_n^\ell(D)$.

Conversely, take one segmentation $\tau \in \mathcal{S}_n^\ell(D)$. Then each segment of τ contains at least one point. Adding $\ell - 1$ points to each segment (from the left edge) clearly provides a segmentation with D segments of length at least ℓ. This allows to conclude. □

This leads to the following generalized change-points detection procedure involving a constraint on the minimum length $\ell \geq 1$ of each segment.

Step 1: Solve

$$\forall\ 1 \leq D \leq D_{\max}, \quad \hat{\tau}_D^\ell = \underset{\tau \in \mathcal{T}_n^\ell(D)}{\arg\min} \|Y - \hat{\mu}^\tau\|_{\mathcal{H},n}^2 , \tag{9}$$

where $\mathcal{T}_n^\ell(D)$ denotes the set of segmentations with D segments of length at least $\ell \geq 1$.

Step 2: Find

$$\hat{D}^\ell = \underset{D}{\arg\min} \left\{ \left\| Y - \hat{\mu}^{\hat{\tau}_D^\ell} \right\|_{\mathcal{H},n}^2 + \mathrm{pen}_\ell(\hat{\tau}_D^\ell) \right\} \qquad \text{and} \quad \hat{\tau}^\ell = \hat{\tau}_{\hat{D}^\ell}^\ell, \tag{10}$$

where $\mathrm{pen}_\ell(\tau) = c_1 D_\tau + c_2 \log \binom{n - D_\tau(\ell-1)-1}{D_\tau - 1}$.

Let us emphasize that this generalized procedure, including this additional parameter $\ell \geq 1$, is very similar the previous one. The optimization step of Eq. (9) is performed by dynamic programming up to a minor change of implementation. The optimization of the second step (10) remains unchanged except it involves a slightly different penalty shape. The tuning of the constant c_1 and c_2 is still made by the slope heuristic (see Section 4).

2.6. A link between kernels and energy-based distances

Note that the kernel-based framework developed in Sections 2.1–2.3 is very general. Various existing procedures can be rephrased in this framework by use of a particular kernel. For example the procedure of [37], which is devoted to the detection of changes in the mean of a one-dimensional real-valued signal, reduces to ours by use of the linear kernel. More interestingly the procedure called ECP developed by [40] and that relies on an energy-based distance to detect changes in multivariate distributions, can also be integrated into our framework using a particular kernel as explained in what follows.

For every $\alpha \in (0, 2)$, let us define $\rho_\alpha(x, y) = \|x - y\|^\alpha$, where $x, y \in \mathbb{R}^q$ and $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^q$. Then ρ_α is a semimetric of negative type [10], and for any independent random variables $X, X', Y, Y' \in \mathbb{R}^d$ with respective probability distributions satisfying $P_X = P_{X'}$ and $P_Y = P_{Y'}$, [40] introduce the energy-based distance:

$$\mathcal{E}(X, Y; \alpha) = 2E\left[\,\rho_\alpha(X, Y)\,\right] - E\left[\,\rho_\alpha(X, X')\,\right] - E\left[\,\rho_\alpha(Y, Y')\,\right], \tag{11}$$

with the assumption that $\max\left(E\left[\,\rho_\alpha(X, X')\,\right], E\left[\,\rho_\alpha(X, Y)\,\right], E\left[\,\rho_\alpha(Y, Y')\,\right]\right) < +\infty$. Then following [48] and for every $x_0 \in \mathbb{R}^d$, we define

$$k_\alpha^{x_0}(x, y) = \frac{1}{2}\left[\,\rho_\alpha(x, x_0) + \rho_\alpha(y, x_0) - \rho_\alpha(x, y)\,\right], \tag{12}$$

which is a positive semi-definite kernel leading to an RKHS $\mathcal{H}_\alpha^0$. Plugging this in Eq. (11), one

can easily check that

$$\begin{aligned}
\mathcal{E}(X,Y;\alpha) &= 2E\left[\,\rho_\alpha(X,x_0)+\rho_\alpha(Y,x_0)-2k_\alpha^{x_0}(X,Y)\,\right] \\
&\quad - E\left[\,\rho_\alpha(X,x_0)+\rho_\alpha(X',x_0)-2k_\alpha^{x_0}(X,X')\,\right] \\
&\quad - E\left[\,\rho_\alpha(Y,x_0)+\rho_\alpha(Y',x_0)-2k_\alpha^{x_0}(Y,Y')\,\right] \\
&= 2E\left[\,k_\alpha^{x_0}(X,X')+k_\alpha^{x_0}(Y,Y')-2k_\alpha^{x_0}(X,Y)\,\right] \\
&= 2\left\|\mu_{P_X}^\alpha-\mu_{P_Y}^\alpha\right\|_{\mathcal{H}_\alpha^0}^2,
\end{aligned}$$

where $\mu_{P_X}^\alpha, \mu_{P_Y}^\alpha \in \mathcal{H}_\alpha^0$ respectively denote the mean elements of the distributions P_X and P_Y, and $\|\cdot\|_{\mathcal{H}_\alpha^0}$ is the norm in $\mathcal{H}_\alpha^0$.

An important consequence of this derivation is that the exact and approximation algorithms described in Section 3 immediately apply to procedures relying on the optimization of the energy-based distance $\mathcal{E}(X,Y;\alpha)$. This is all the more noticeable as our *exact algorithm* has a lower memory complexity than the *approximate optimization algorithm* of ECP (with a similar time complexity). Therefore, for the same computational time, our exact optimization algorithm could replace the approximate algorithm used in ECP. One can also emphasize that our approximate algorithm which is even faster could also be used (see its description in Section 3.2). This particular energy-based kernel, which is a characteristic kernel, has been involved in our simulation experiments as well (Section 4.2.1).

3. New algorithms

In this section we first show how to avoid the preliminary calculation of the cost matrix required by [28] to apply dynamic programming. The key idea is to compute the elements of the cost matrix on the fly when they are required by the dynamic programming algorithm. Roughly, this can be efficiently done by reordering the loops involved in Algorithm 1 proposed in [28]. This leads to the new exact Algorithm 3. It has a reduced space complexity of order $\mathcal{O}(n)$ compared to $\mathcal{O}(n^2)$ for the one used in [28]. Note that including the constraint (introduced in Section 2.5) on the segment sizes mostly change the index of the **for** loops in Algorithm 3. We choose to describe the algorithm in the unconstrained version (7) to ease the understanding.

Second, we provide a faster but approximation algorithm (Section 3.2), which enjoys a smaller complexity of order $\mathcal{O}(D_{\max}n)$ in time. It combines a low-rank approximation to the Gram matrix and the use of the binary segmentation heuristic (Section 3.2.2). This approximation algorithm allows the analysis of very large signals ($n \geq 10^6$).

3.1. New efficient algorithm to recover the best segmentation from the Gram matrix

As exposed in Section 2.4, the main computational cost of the change-point detection procedure results from Eq. (7), that is recovering the best segmentation with $1 \leq D \leq D_{\max}$ segments and

solving

$$\mathbf{L}_{D,n+1} = \min_{\tau \in \mathcal{T}_D} \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} \qquad \text{(best fit to the data)}$$

$$\hat{m}_D = \arg\min_{\tau \in \mathcal{T}_D} \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} \qquad \text{(best segmentation)} \tag{13}$$

for every $1 \leq D \leq D_{\max}$, where $\mathcal{T}_D$ denotes the collection of segmentations of $\{1, \ldots, n\}$ with D segments. This challenging step involves the use of dynamic programming [9, 7], which provides the exact solution to the optimization problem (13). Let us first provide some details on the usual way dynamic programming is implemented.

3.1.1. Limitations of the standard dynamic programming algorithm for kernels

Let τ denote a segmentation in D segments (with the convention that $\tau_1 = 1$ and $\tau_{D+1} = n+1$). For any $1 \leq d \leq D$, the segment $\{\tau_d, \ldots, \tau_{d+1} - 1\}$ of the segmentation τ has a cost that is equal to

$$C_{\tau_d, \tau_{d+1}} = \sum_{i=\tau_d}^{\tau_{d+1}-1} k(X_i, X_i) \; - \; \frac{1}{\tau_{d+1} - \tau_d} \sum_{i=\tau_d}^{\tau_{d+1}-1} \sum_{j=\tau_d}^{\tau_{d+1}-1} k(X_i, X_j). \tag{14}$$

Then the cost of the segmentation τ is given by

$$\|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} = \sum_{d=1}^{D} C_{\tau_d, \tau_{d+1}},$$

which is clearly *segment additive* [28, 3].

Dynamic programming solves (13) for all $1 \leq D \leq D_{\max}$ by applying the following update rules

$$\forall\ 2 \leq D \leq D_{\max}, \quad \mathbf{L}_{D,n+1} = \min_{\tau \leq n} \{\ \mathbf{L}_{D-1,\tau} + C_{\tau,n+1}\ \}, \tag{15}$$

which exploits the property that the optimal segmentation in D segments over $\{1, \ldots, n\}$ can be computed from optimal ones with $D-1$ segments over $\{1, \ldots, \tau\}$ ($\tau \leq n$). Making the key assumption that *the cost matrix* $\{C_{i,j}\}_{1 \leq i,j \leq n+1}$ *has been stored*, we can compute $\mathbf{L}_{D,n+1}$ with Algorithm 1.

Algorithm 1 Basic use of Dynamic Programming

1: **for** $D = 2$ to $D_{\max}$ **do**
2: **for** $\tau' = D$ to n **do**
3: $\mathbf{L}_{D,\tau'+1} = \min_{\tau \leq \tau'} \{\ \mathbf{L}_{D-1,\tau} + C_{\tau,\tau'+1}\ \}$
4: **end for**
5: **end for**

This algorithm is used by [28] and suffers two main limitations. First it assumes that the $C_{\tau,\tau'}$ have been already computed, and does not take into account the computational cost of its calculation. Second, it stores all $C_{\tau,\tau'}$ in a $\mathcal{O}(n^2)$ matrix, which is memory expensive.

A quick inspection of the algorithm reveals that the main step at Line 3 requires $\mathcal{O}(\tau')$ operations (assuming the $C_{i,j}$s have been already computed). Therefore, with the two **for** loops we get

a complexity of $\mathcal{O}(D_{\max} n^2)$ in time. Note that without any particular assumption on the kernel $k(\cdot, \cdot)$, computing $\|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n}$ for a given segmentation τ is already of order $\mathcal{O}(n^2)$ in time since it involves summing over a quadratic number of terms of the Gram matrix (see Eq. (14)). Therefore, there is no hope to solve (13) exactly in less than quadratic time without additional assumptions on the kernel.

From Eq. (14) let us also remark that computing each $C_{i,j}$ $(1 \leq i < j \leq n)$ naively requires itself a quadratic number of operations. Computing the whole cost matrix would require a complexity $\mathcal{O}(n^4)$ in time. Taking this into account, the dynamic programming step (Line 3 of Algorithm 1) is not the limiting factor and the overall time complexity of Agorithm 1 is $\mathcal{O}(n^4)$.

Finally, let us emphasize that this high computational burden is not specific of detecting change-points with kernels. It is rather representative of most learning procedures based on reproducing kernels and the associated Gram matrix [8].

3.1.2. Improved use of dynamic programming for kernel methods

Reducing space complexity. From Algorithm 1, let us first remark that each $C_{\tau,\tau'}$ is used several times along the algorithm. A simple idea to avoid that is to swap the two **for** loops in Algorithm 1. This leads to the following modified Algorithm 2, where each column $C_{\cdot,\tau'+1}$ of the cost matrix is only used once unlike in Algorithm 1.

Algorithm 2 Improved space complexity

1: **for** $\tau' = 2$ to n **do**
2: **for** $D = 2$ to $\min(\tau', D_{\max})$ **do**
3: $\mathbf{L}_{D,\tau'+1} = \min_{\tau \leq \tau'} \{ \mathbf{L}_{D-1,\tau} + C_{\tau,\tau'+1} \}$
4: **end for**
5: **end for**

Importantly swapping the two **for** loop does not change the output of the algorithm and does not induce any additional calculations. Furthermore, at step τ' of the first **for** loop we do not need the whole $n \times n$ cost matrix to be stored, but only the column $C_{\cdot,\tau'+1}$ of the cost matrix. This column is of size at most $\mathcal{O}(n)$.

Algorithm 2 finally requires storing coefficients $\{\mathbf{L}_{d,\tau}\}_{1 \leq d \leq D,\ 2 \leq \tau \leq n}$ that are computed along the algorithm as well as successive column vectors $\{C_{\cdot,\tau}\}_{2 \leq \tau \leq n}$ (of size at most n) of the cost matrix. This leads to an overall complexity of $\mathcal{O}(D_{\max} n)$ in space. The only remaining problem is to compute these successive column vectors efficiently. Let us recall that a naive implementation is prohibitive: each coefficient of the column vector can be computed in $\mathcal{O}(n^2)$, which would lead to $\mathcal{O}(n^3)$ to get the entire column.

Iterative computation of the columns of the cost matrix. The last ingredient of our final exact algorithm is the efficient computation of each column vector $\{C_{\cdot,\tau}\}_{2 \leq \tau \leq n}$. Let us explain how to iteratively compute each vector in linear time.

First it can be easily observed that Eq. (14) can be rephrased as follows

$$C_{\tau,\tau'} = \sum_{i=\tau}^{\tau'-1} \left(k\left(X_i, X_i\right) - \frac{A_{i,\tau'}}{\tau' - \tau} \right) = D_{\tau,\tau'} - \frac{1}{\tau' - \tau} \sum_{i=\tau}^{\tau'-1} A_{i,\tau'},$$

where $D_{\tau,\tau'} = \sum_{i=\tau}^{\tau'-1} k\left(X_i, X_i\right)$ and

$$A_{i,\tau'} = -k(X_i, X_i) + 2 \sum_{j=i}^{\tau'-1} k(X_i, X_j).$$

Second, both $D_{\tau,\tau'}$ and $\{A_{i,\tau'}\}_{i \leq \tau'}$ can be iteratively computed from τ' to $\tau' + 1$ by use of the two following equations:

$$D_{\tau,\tau'+1} = D_{\tau,\tau'} + k(X_{\tau'}, X_{\tau'}), \qquad \text{and} \qquad A_{i,\tau'+1} = A_{i,\tau'} + 2k(X_{\tau'}, X_{\tau'}),\ \forall i \leq \tau',$$

with $A_{\tau'+1,\tau'+1} = -k(X_{\tau'+1}, X_{\tau'+1})$. Therefore, as long as computing $k(x_i, x_j)$ requires $\mathcal{O}(1)$ operations, updating from τ' to $\tau' + 1$ requires $\mathcal{O}\left(\tau'\right)$ operations.

Remark 3. *Note that for many classical kernels, computing $k(x_i, x_j)$ is indeed $\mathcal{O}(1)$ in time. If $x_i \in \mathbb{R}^q$ with q a positive integer being negligible with respect to other influential quantities such as $D_{\max}$ and n, several kernels such as the Gaussian, Laplace, or χ^2 ones lead to a $\mathcal{O}(q) = \mathcal{O}(1)$ time complexity for evaluating $k(x_i, x_j)$. By contrast in case where q is no longer negligible, the resulting time complexity is multiplied by a factor q, which corroborates the intuition that the computational complexity increases with the "complexity" of the objects in $\mathcal{X}$.*

This update rule leads us to the following Algorithm 3, where each column $C_{\cdot,\tau'+1}$ in the first **for** loop is computed only once:

Algorithm 3 Improved space and time complexity (*Kernseg*)

1: **for** $\tau' = 2$ to n **do**
2: Compute the $(\tau' + 1)$-th column $C_{\cdot,\tau'+1}$ from $C_{\cdot,\tau'}$
3: **for** $D = 2$ to $\min(\tau', D_{\max})$ **do**
4: $\mathbf{L}_{D,\tau'+1} = \min_{\tau \leq \tau'} \{\mathbf{L}_{D-1,\tau} + C_{\tau,\tau'+1}\}$
5: **end for**
6: **end for**

From a computational point of view, each step of the first **for** loop in Algorithm 3 requires $\mathcal{O}(\tau')$ operations to compute $C_{\cdot,\tau'+1}$ and at most $\mathcal{O}(D_{\max}\tau')$ additional operations to perform the dynamic programming step at Line 4. Then the overall complexity is $\mathcal{O}(D_{\max}n^2)$ in time and $\mathcal{O}(D_{\max}n)$ in space. This should be compared to the $\mathcal{O}(D_{\max}n^4)$ time complexity of the naive calculation of the cost matrix and to the $\mathcal{O}(n^2)$ space complexity of the standard Algorithm 1 from [28].

3.1.3. Runtimes comparison to other implementations

The purpose of the present section is to perform the comparison between Algorithm 3 and other competitors to illustrate their performances as the sample size increases with $D_{\max} = 100$.

The first comparison has been carried out between Algorithm 3 and the naive quartic computation of the cost matrix (Algorithm 1). These two algorithms have been implemented in C and packaged in R. Results for these algorithms are reported in Figure 1 (Left). Unsurprisingly, our quadratic algorithm (called *Kernseg*) is faster than a quartic computation of the cost matrix (called KCP) even for very small sample sizes ($n < 320$).

Second, we also compared the runtime of *Kernseg* (Algorithm 3) with that of ECP discussed in Section 2.6 implemented in the R-package [32] (see the middle panel of Figure 1). Since ECP is based on the binary segmentation heuristic applied to an energy-based distance, its worst-case complexity is at most $\mathcal{O}(D_{\max} n^2)$ in time, which is the same as that of *Kernseg*. Note also that the native implementation of ECP involves an additional procedure relying on permutations to choose the number of change-points. If B denotes the number of permutations, the induced complexity is then $\mathcal{O}(BD_{\max} n^2)$ in time. To be fair, we compared our approach and ECP with and without the permutation layer. Finally it is also necessary to emphasize that unlike *Kernseg*, ECP does not provide the exact but only an approximate solution to the optimization problem (13). Results are summarized in Figure 1 (Middle). It illustrates that our exact algorithm (*Kernseg*) has a quadratic complexity similar to that of ECP with and without permutations. Our algorithm is the overall fastest one even for small sample size ($n < 1\,000$). Although this probably results from implementation differences, it is still noteworthy since *Kernseg* is exact unlike ECP.

Finally, Figure 1 (Right) illustrates the worse memory use of ECP (with and without any permutations) as compared to that of the exact KS.Gau (*Kernseg* used with the Gaussian kernel). ECP has an $\mathcal{O}(n^2)$ space complexity, while KS.Gau is $\mathcal{O}(n)$. For n larger than 10^4 the quadratic space complexity of ECP is a clear limitation since several Gb of RAM are required.

3.2. Approximating the Gram matrix to speed up the algorithm

In Section 3.1.2, we described an improved algorithm called *Kernseg*, which carefully combines dynamic programming with the computation of the cost matrix elements. This new algorithm (Algorithm 3) provides the exact solution to the optimization problem given by Eq. (13). However without any further assumption on the underlying reproducing kernel, this algorithm only achieves the complexity $\mathcal{O}(n^2)$ in time, which is a clear limitation with large scale signals ($n \geq 10^5$). Note also that this limitation results from the use of general positive semi-definite kernels (and related Gram matrices) and cannot be improved by existing algorithms to the best of our knowledge. For instance, the binary segmentation heuristic [22], which is known to be computationally efficient for parametric models, suffers the same $\mathcal{O}(n^2)$ time complexity when used in the reproducing kernel framework (see Section 3.2.2).

Let us remark however that for some particular kernels it is possible to reduce this time complexity. For example for the linear one $k(x, y) = \langle x, y \rangle_{\mathbb{R}^d}$, $x, y \in \mathbb{R}^d$, one can use the following

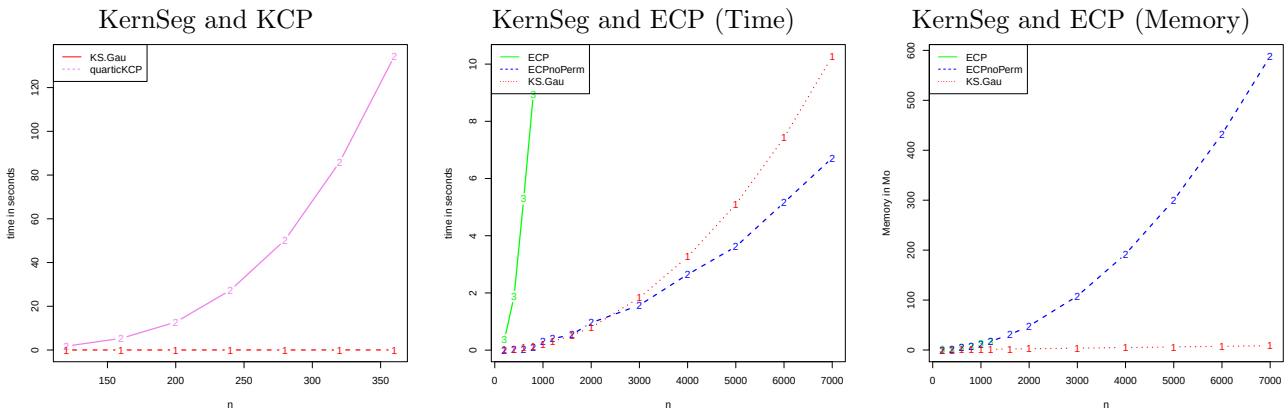

Figure 1: (Left) Runtime in seconds of Algorithm 3 as a function of the length of the signal (n) for $D_{\max} = 100$. (1-red) and a quartic computation of the cost matrix (2-violet). (Middle) Runtime in seconds of Algorithm 3 as a function of the length of the signal (1-red) and of ECP without permutation (2-blue) and ECP with the default number of permutations (3-green). (Right) Memory in mega-octet of Algorithm 3 as a function of the length of the signal (1-red) and of ECP without permutation (2-blue) and ECP with the default number of permutations (3-green). The performances of ECP with or without permutation are exactly the same.

trick

$$\sum_{1\leq i\neq j\leq n} k(X_i, X_j) = \sum_{1\leq i\leq n} \left\langle X_i, \sum_{j=1}^{n} X_j - X_i \right\rangle = \left\| \sum_{i=1}^{n} X_i \right\|^2 - \sum_{i=1}^{n} \|X_i\|^2 , \tag{16}$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^d$.

The purpose of the present section is to describe a versatile strategy (*i.e* applicable to any kernel) relying on a low-rank approximation to the Gram matrix [52, 49, 21]. This approximation allows to considerably reduce the computation time by exploiting (16). Note however that the resulting procedure achieves this lower time complexity at the price of only providing an approximation to the exact solution to (13) (unlike the algorithm described in Section 3.1.2).

3.2.1. Low-rank approximation to the Gram matrix

The main idea is to follow the same strategy as the one described by [20] to derive a low-rank approximation to the Gram matrix $\mathbf{K} = \{\mathbf{K}_{i,j}\}_{1\leq i,j\leq n}$, where $\mathbf{K}_{i,j} = k(X_i, X_j)$.

Assuming $\mathbf{K}$ has rank $\mathrm{rk}(\mathbf{K}) \ll n$, we could be tempted to compute the best rank approximation to $\mathbf{K}$ by computing the $\mathrm{rk}(\mathbf{K})$ largest eigenvalues (and corresponding eigenvectors) of $\mathbf{K}$. However such computations induce a $\mathcal{O}(n^3)$ time complexity which is prohibitive.

Instead, [20] suggest applying this idea on a square sub-matrix of $\mathbf{K}$ with size $p \ll n$. For any subsets $I, J \subset \{1, \ldots, n\}$, let $\mathbf{K}_{I,J}$ denote the sub-Gram matrix with respectively row and column indices in I and J. Let $J_p \subset \{1, \ldots, n\}$ denote such a subset with cardinality p, and consider the sub-Gram matrix $\mathbf{K}_{J_p,J_p}$ which is of rank $r \leq p$. Further assuming $r = p$, the best rank p approximation to $\mathbf{K}_{J_p,J_p}$ is $\mathbf{K}_{J_p,J_p}$ itself. This leads to the final approximation to the Gram Matrix

$\mathbf{K}$ [20, 8] by

$$\widetilde{\mathbf{K}} = \mathbf{K}_{I_n,J_p}\, \mathbf{K}^{+}_{J_p,J_p}\, \mathbf{K}_{J_p,I_n},$$

where $I_n = \{1,\dots,n\}$, and $\mathbf{K}^{+}_{J_p,J_p}$ denotes the pseudo-inverse of $\mathbf{K}_{J_p,J_p}$. Further considering the SVD decomposition of $\mathbf{K}_{J_p,J_p} = \mathbf{U}'\Lambda\mathbf{U}$, for an orthonormal matrix $\mathbf{U}$, we can rewrite

$$\widetilde{\mathbf{K}} = \mathbf{Z}'\mathbf{Z}, \qquad \text{with} \quad \mathbf{Z} = \Lambda^{-1/2}\mathbf{U}\, \mathbf{K}_{J_p,I_n} \in \mathcal{M}_{p,n}(\mathbb{R}).$$

Note that the resulting time complexity is $\mathcal{O}(p^2 n)$, which is smaller than the former $\mathcal{O}(n^3)$ as long as $p = o(\sqrt{n})$. This way, columns $\{Z_i\}_{1\leq i\leq n}$ of $\mathbf{Z}$ act as new p-dimensional observations, and each $\widetilde{\mathbf{K}}_{i,j}$ can be seen as the inner product between two vectors of $\mathbb{R}^p$, that is

$$\widetilde{\mathbf{K}}_{i,j} = Z_i' Z_j. \tag{17}$$

The main interest of this approximation is that, using Eq. (16), computing the cost of a segment of length t has a complexity $\mathcal{O}(t)$ in time unlike the usual $\mathcal{O}(t^2)$ that holds with general kernels.

Interestingly such an approximation to the Gram matrix can be also built from a set of deterministic points in $\mathcal{X}$. This remark has been exploited to compute our low-rank approximation for instance in the simulation experiments as explained in Section 4.4.5.

Note that choosing the set J_p of columns/rows leading to the approximation $\widetilde{\mathbf{K}}$ is of great interest in itself for at least two reasons. First from a computational point of view, the p columns have to be selected following a process that does not require to compute the n possible columns beforehand (which would induce an $O(n^2)$ time complexity otherwise). Second, the quality of $\widetilde{\mathbf{K}}$ to approximate $\mathbf{K}$ crucially depends on the rank of $\widetilde{\mathbf{K}}$ that has to be as close as possible to that of $\mathbf{K}$, which remains unknown for computational reasons. However such questions are out of scope of the present paper, and we refer interested readers to [52, 20, 8] where this point has been extensively discussed.

3.2.2. Binary segmentation heuristic

Since the low-rank approximation to the Gram matrix detailed in Section 3.2.1 leads to finite dimensional vectors in $\mathbb{R}^p$ (17), the change-point detection problem described in Section 2.3 amounts to recover abrupt changes of the mean of a p-dimensional time-series. Therefore any existing algorithm usually used to solve this problem in the p-dimensional framework can be applied. An exhaustive review of such algorithms is out of the scope of the present paper. However we will mention only a few of them to highlight their drawbacks and motivate our choice. Let us also recall that our purpose is to provide an efficient algorithm allowing: (i) to (approximately) solve Eq. (13) for each $1 \leq D \leq D_{\max}$ and (ii) to deal with large sample sizes ($n \geq 10^6$).

The first algorithm is the usual version of constrained dynamic programming [7]. Although it has been recently revisited with $p = 1$ by [46, 16, 39], it has a $\mathcal{O}(n^2)$ time complexity with $p > 1$, which excludes dealing with large sample sizes. Another version of regularized dynamic

programming has been explored by [34] who designed the PELT procedure. It provides the best segmentation over all segmentations with a penalty of λ per change-point with an $\mathcal{O}(n)$ complexity in time if the number of change-points is linear in n. Importantly, the complexity of the pruning inside PELT depends on the true number of change-points. For only a few change-points, the PELT complexity remains quadratic in time. With PELT, it is not straightforward to efficiently solve Eq. (13) for each $1 \leq D \leq D_{\max}$, which is precisely the goal we pursue. Note however that it would still be possible to recover some of those segmentations by exploring a range of λ values like in CROPS [30].

A second possible algorithm is the so-called *binary segmentation* [43, 53, 22] that is a standard heuristic for approximately solving Eq. (13) for each $1 \leq D \leq D_{\max}$. This iterative algorithm computes the new segmentation $\widetilde{\tau}(D+1)$ with $D+1$ segments from $\widetilde{\tau}(D)$ by splitting one segment of $\widetilde{\tau}(D)$ into two new ones without modifying other segments. More precisely considering the set of change-points $\widetilde{\tau}(D) = \{\tau_1, \ldots, \tau_{D+1}\}$, binary segmentation provides

$$\widetilde{\tau}(D+1) = \underset{\tau \in \mathcal{T}_{D+1} | \tau \cap \widetilde{\tau}(D) = \widetilde{\tau}(D)}{\arg\min} \left\{ \|Y - \hat{\mu}^{\tau}\|^2_{\mathcal{H},n} \right\}.$$

Since only one segment of the previous segmentation is divided into two new segments at each step, the binary segmentation algorithm provides a simple (but only approximate) solution to Eq. (13) for each $1 \leq D \leq D_{\max}$.

We provide some pseudo-code for binary segmentation in Algorithm 5. It uses a sub-routine described by Algorithm 4 to compute the best split of any segment $[\tau, \tau'[$ of the data. To be specific, this BestSplit routine outputs four things: (1) the reduction in cost of spliting the segment $[\tau, \tau'[$, (2) the best change to split (3) the resulting left segment and (4) the resulting right segment.

In the binary segmentation algorithm candidate splits are stored and handled using a binary heap data structure [18] using the reduction in cost as a key. This data structure allows to efficiently insert new splits and extract the best split in $\mathcal{O}(\log(D_{max}))$ at every time step. Without such a structure inserting splits and extracting the best split would typically be in $\mathcal{O}(D_{max})$ and for large D_{max} the binary segmentation heuristic is at best $\mathcal{O}(n^2)$. Note that the RBS procedure [45], which is involved in our simulation experiments (Section 4.2.3), also uses this heuristic.

Algorithm 4 BestSplit of segment $[\tau, \tau'[$

1: $\hat{m} = \min_{\tau < t < \tau'} \{C_{\tau,t} + C_{t,\tau'}\}$ and $\hat{t} = \arg\min_{\tau < t < \tau'} \{C_{\tau,t} + C_{t,\tau'}\}$

2: Output four things (1) $C_{\tau,t} - \hat{m}$, (2) $\hat{t}$, (3) $[\tau, \hat{t}[$ and (4) $[\hat{t}, \tau'[$

Algorithm 5 Binary Segmentation

Segs $= \{[1, n+1[\}$
Changes $= \emptyset$
CandidateSplit $= \emptyset$ [a binary heap]
for D_{max} iteration **do**
  for $aseg \in Segs$ **do**
    Insert BestSplit($aseg$) in CandidateSplit
  end for
  Extract the best split of CandidateSplit and recover: $\hat{t}$, $[\tau, \hat{t}[$ and $= [\hat{t}, \tau'[$
  Add $\hat{t}$ in Changes
  Set Segs to $\{ [\tau, \hat{t}[, [\hat{t}, \tau'[\}$
end for

Assuming the best split of any segment is linear in its length the overall time complexity of binary segmentation for recovering approximate solutions to (13) for all $1 \leq D \leq D_{\max}$ is around $\mathcal{O}(\log(D_{\max})n)$ in practice. The worst case time complexity is $\mathcal{O}(D_{\max}n)$. A typical setting where it is achieved is with the linear kernel when $i \mapsto X_i = \exp(i)$ for instance. At the i-th iteration of the binary segmentation algorithm, the best split of a segment of length $n - i + 1$ corresponds to one segment of length 1 and another one of length $n - i$.

An important remark is that binary segmentation only achieves this reduced $\mathcal{O}(\log(D_{\max})n)$ time complexity provided that recovering the best split of any segment is linear in its length. This is precisely what has been allowed by the low-rank matrix approximation summarized by Eq. (17). Indeed with the low-rank approximation, computing the best split of any segment is linear in n and p. The resulting time complexity of binary segmentation is thus $\mathcal{O}(p\log(D_{\max})n)$, which reduces to $\mathcal{O}(\log(D_{\max})n)$ as long as p is small compared to n. By contrast without the approximation recovering the best split is typically quadratic in the length of the segment and binary segmentation would suffer an overall time complexity of order $\mathcal{O}(\log(D_{\max})n^2)$ or $\mathcal{O}(D_{\max}n^2)$.

3.2.3. Implementation and runtimes of the approximate solution

The approximation algorithm we recommend is the combination of the low-rank approximation step detailed in Section 3.2.1 and of the binary segmentation discussed in Section 3.2.2. The resulting time complexity is then $\mathcal{O}(p^2n + p\log(D_{\max})n)$, which allows dealing with large sample sizes ($n \geq 10^6$).

From this time complexity it arises that an influential parameter is the number p of columns of the matrix used to build the low-rank approximation. In particular this low-rank approximation remains computationally attractive as long as $p = o(\sqrt{n})$. Figure 2 illustrates the actual time complexity of this fast algorithm (implemented in C) with respect to n for various values of p: (i) a constant value of p and (ii) $p = \sqrt{n}$. To ease the comparison, we also plotted the runtime of the exact algorithm (Algorithm 3) detailed in Section 3.1.2 and RBS that uses binary segmentation

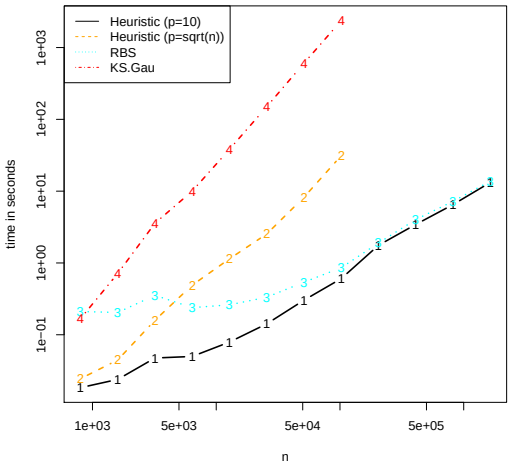

Figure 2: Runtime as a function of n (length of the signal) for $D_{max} = 100$. Runtime of our approximation algorithm with $p = 100$ (1-black) and $p = \sqrt{n}$ (2-orange), RBS (3-cyan), exact Algorithm 3 (4-red)

(see Section 4.2.3).

Our fast approximation algorithm ($ApKS$) recovers a quadratic complexity if $p = \sqrt{n}$. However its overhead is much smaller than that of the exact algorithm, which makes it more applicable than the latter with large signals in practice. Note also that Figure 2 illustrates that $ApKS$ returns the solution in a matter of seconds with a sample size of $n = 10^5$, which is much faster than *Kernseg* (based on dynamic programming) that requires a few minutes. The RBS implementation involves preliminary calculations which make it slower than $ApKS$ with $n \leq 2 \cdot 10^3$. However for larger values ($n \geq 10^4$) RBS is as fast as $ApKS$ with $p = 10$.

4. Segmentation assessment

From a statistical point of view *Kernseg* provides the same performance as that of [3]. However it greatly improves on the latter in terms of computational complexity as proved in Section 3.1. In their simulation experiments [3] mainly focus on detecting change-points in the distribution of $\mathbb{R}$-valued data as well as of more structured objects such as histograms. Here we rather investigate the performance of the kernel change-point procedure on specific two-dimensional biological data: the DNA copy number and the BAF profiles (see Section 4.1.1). More precisely our experiments highlight two main assets of applying reproducing kernels to these biological data: (i) reasonable kernels avoid the need for modeling the type of change-points we are interested in and improve upon state-of-the-art approaches in this biological context, and (ii) the high flexibility of kernels facilitates data fusion, that is allows to combine different data-types and get more power to detect true change-points.

In the following we first briefly introduce the type of data we are looking at, and describe our

simulation experiments obtained by resampling from a set of real annotated DNA profiles. Second, we provide details about the change-point procedures involved in our comparison. We also define the criteria used to assess the performance of the estimated segmentations. Finally, we report and discuss the results of these experiments.

4.1. Data description

4.1.1. DNA copy number data

DNA copy number alterations are a hallmark of cancer cells [27]. The accurate detection and interpretation of such changes are two important steps toward improved diagnosis and treatment. Normal cells have two copies of DNA, inherited from each biological parent of the individual. In tumor cells, parts of a chromosome of various sizes (from kilobases to a chromosome arm) can be deleted, or copied several times. As a result, DNA copy numbers in tumor cells are piecewise constant along the genome. Copy numbers can be measured using microarray or sequencing experiments. Figure 3 displays an example of copy number profiles that can be obtained from SNP-array data [42].

The left panel (denoted by TCN) represents estimates of the total copy number (TCN). The right panel (denoted by BAF) represents estimates of allele B fractions (BAF) using only homozygous position. We refer to [42] for an explanation of how these estimates are obtained. In the normal region [0-2200], TCN is centered around two copies and BAF has three modes at 0, 1/2 and 1.

On top of Figure 3, numbers (a, b) represent each parental copy number in the corresponding segment. For instance $(0, 2)$ means that the total number of copies in the segment is 2. But a copy from one of the two parents is missing while the other copy has been duplicated. Importantly any

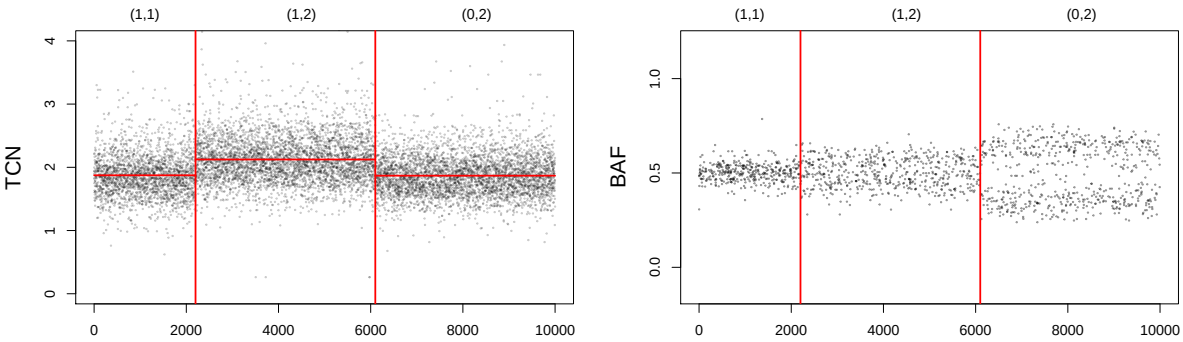

Figure 3: SNP array data. Total copy numbers (TCN), allelic ratios (BAF) along 10,000 genomic loci. Red vertical lines represent change-points, and red horizontal lines represent estimated mean signal levels between two change-points.

change in only one of the parental copy numbers is reflected in both TCN and BAF. Therefore it makes sense to jointly analyze both dimensions to ease the identification of change-points.

Importantly in the following, allelic ratios (BAF) are always symmetrized (or folded)– that is we consider $|BAF - 0.5|$–to facilitate the segmentation task. This is common practice in the field [51].

4.1.2. Generated data

Realistic DNA profiles with known truth (similar to that of Figure 3) have been generated using the `acnr` package [45]. The constituted benchmark consists of profiles with $5,000$ positions of heterozygous SNPs and exactly $K = 10$ change-points. As in [45] we only consider four biological states for the segments. The `acnr` package allows to vary the difficulty level by adding normal cell contamination, thus degrading tumor percentage. Three levels of difficulty have been considered by varying tumor percentage: 100% (easy case), 70%, and 50% (difficult case). Figure 4 displays three examples of simulated profiles (one for each tumor purity level).

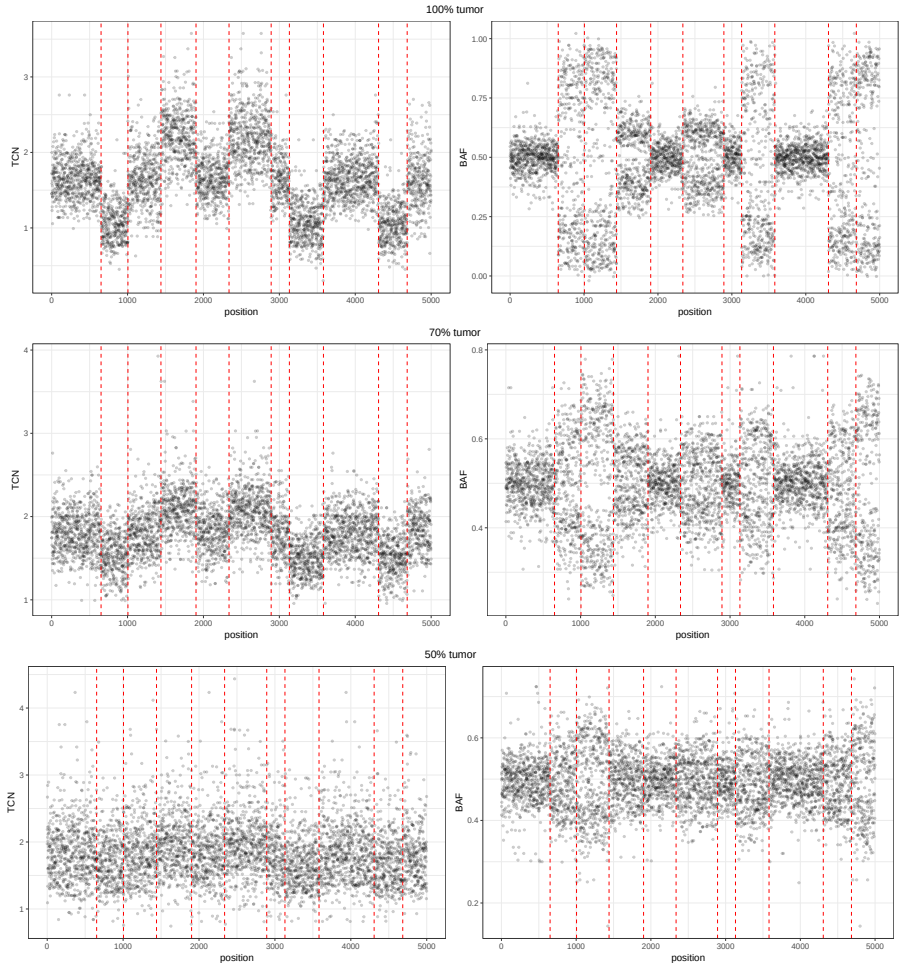

Figure 4: Benchmark 1: Profiles simulated with the `acnr` package. Each line corresponds to a tumor percentage (100%, 70% and 50%). The first column correspond to copy number (TCN) and the second to the allele B fraction (BAF).

For each level, $N = 50$ profiles (with the same segment states and change-points) are generated making a total of 150 simulated profiles both for BAF and TCN.

4.2. Competing procedures

4.2.1. Kernseg *and* ApKS

The implemented exact algorithm *Kernseg* (corresponding to Algorithm 3) and its fast approximation *ApKS* (based on binary segmentation) are applicable with any kernel (Gaussian, exponential, polynomial, ...).

In our simulation experiments we consider three kernels.

- The first one is the so-called linear kernel defined by $k(x, y) = \langle x, y \rangle$, where $x, y \in \mathbb{R}$. It is used as a benchmark since *Kernseg* with this kernel reduces to the procedure of [37]. The corresponding procedure is denoted by KS.Lin.

- The second one is the Gaussian kernel defined for every $x, y \in \mathbb{R}$ by

$$k_\delta(x, y) = \exp\left[\frac{-|x-y|^2}{\delta}\right], \qquad \forall \delta > 0.$$

  Since it belongs to the class of characteristic kernels [50], it is a natural choice to detect any abrupt changes arising in the full distribution [3]. We call this procedure KS.Gau.

- The third one is the kernel associated with the energy-based distance introduced in Eq. (12) with $\alpha = 1$ and $x_0 = 0$. This particular choice is the prescribed value in the ECP package [32]. We call this procedure KS.ECP.

These three kernels allow to (i) illustrate the interest of characteristic kernels compared to non characteristic ones, and (ii) assess the performances of change-point detection with kernels (*Kernseg*) compared to other approaches (ECP, RBS). However other characteristic kernels such as the Laplace or exponential ones (see Section 2.2) could have been considered as well.

For all kernels we considered $D_{max} = 100$. Note also that for all approaches and for both TCN and BAF profiles we first scaled the data using a difference based estimator of the variance. To be specific we get an estimator of the variances by dividing by $\sqrt{2}$ the median absolute deviation of disjoint successive differences. This is common practice in the change-point literature (see for example [22]). Such estimators are less sensitive to any shift in the mean than the classical ones. For the Gaussian kernel we then used $\delta = 1$.

As mentioned earlier, one main asset of kernels is that they allow to easily perform data fusion, which consists here in combining several data profiles to increase the power of detecting small changes arising at the same location in several of them. Here the joint segmentation of the two-dimensional signal (TCN, BAF) is carried out by defining a new kernel as the sum of two coordinate-wise kernels [5], that is

$$k(x_1, x_2) = k(c_1, c_2) + k(b_1, b_2) \tag{18}$$

with $x_1 = (c_1, b_1)$ and $x_2 = (c_2, b_2)$ where the first coordinates of x_1 and x_2 refer to TCN and the second ones to BAF.

Remark 4. *Let us point out that many alternative ways exist to build such a "joint kernel", using the standard machinery of reproducing kernels exposed in [5, 24].*

For instance replacing the sum in Eq. (18) *by a product of kernels is possible. With the Gaussian kernel, this amounts to consider one Gaussian kernel applied to a mixture of squared norms where each coordinate receives a different weight depending on its influence. Another promising direction is to exploit some available side information about the importance of each coordinate in detecting change-points. This can be done by considering a convex sum of kernels where the weights reflect this* a priori *knowledge.*

Finally let us mention that designing the optimal kernel for a learning task is a widely open problem in the literature even if some attempts exist (see Section 7.2 in [2] for a thorough discussion, and [26] for a first partial answer with two-sample tests).

4.2.2. ECP

The ECP procedure [40] (earlier discussed in Section 2.6) has been also introduced in our comparison since it allows us to detect changes in the distribution of multivariate observations.

We used the implementation provided by the authors in the R-package [32]. We used the default parameters $\alpha = 1$ and $\ell = 30$ (minimum length of any segment). Let us notice that, unlike our kernel-based procedures relying on efficiently minimizing a prescribed penalized criterion, ECP chooses the number of segments by iteratively testing each new candidate change-point by means of a permutation test, which makes it highly time-consuming on large profiles (around 15 minutes per profiles for $n = 5000$ compared to 5 seconds for KS.Gau).

4.2.3. Recursive Binary Segmentation (RBS)

In the recent paper by [45], it has been shown that for a known number of change-points the Recursive Binary Segmentation (RBS) [25] is a state-of-the-art change-point procedure for analyzing (TCN, BAF) profiles. RBS is a two-step procedure. In a first step it uses the binary segmentation heuristic (described in Section 3.2.2) on the (TCN) or (TCN,BAF) profile. In a second step it uses dynamic programming on the set of changes identified by the binary segmentation heuristic. We refer interested readers to [45] for a discussion as to why RBS can outperform a pure dynamic programming strategy despite the fact that it provides only an approximation to the targeted optimization problem.

Since the present biological context is the same as that of [45], we therefore decided to carry out the comparison between our kernel-based procedures and RBS.

From a computational perspective RBS relies on the binary segmentation algorithm described in Algorithm 5. The final segmentation output by RBS is then an approximate solution to the optimization problem (in the same way as $ApKS$), while being efficiently computed as illustrated by Figure 2.

4.3. Performance assessment

The quality of the resulting segmentations is quantified in two ways. First we infer the ability of the procedure to provide a reliable estimate of the regression function by computing the quadratic risk of the estimator based on the TCN profile (Section 4.3.1). Second, we also assess the quality of the estimated segmentations by measuring the discrepancy between the true and estimated change-points using the Frobenius distance (Section 4.3.2).

4.3.1. Risk of a segmentation

From a practical point of view, there is no hope to recover true change-points in regions where the signal-to-noise ratio is too low without including false positives, which we would like to avoid. In such non-asymptotic settings, the quality of the estimated segmentation τ can be measured by the risk $R(\hat{f}^\tau)$ which measures the gap between the regression function $f = (f_1, \ldots, f_n) \in \mathbb{R}^n$ and its piecewise-constant estimator based on τ, that is $\hat{f}^\tau = \left(\hat{f}_1^\tau, \ldots, \hat{f}_n^\tau\right) \in \mathbb{R}^n$. This risk is defined by

$$R(\hat{f}^\tau) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left[\left(f_i - \hat{f}_i^\tau\right)^2 \right].$$

In the following simulation results, the risks of all segmentations are always computed with respect to the regression function of the corresponding TCN profile.

4.3.2. Frobenius distance

We also quantify the gap between a segmentation τ and the true segmentation τ^* by using the Frobenius distance [36] between matrices as follows. First, for any segmentation $\tau = (\tau_1, \tau_2, \ldots, \tau_D)$, let us introduce a matrix $M^\tau = \left\{M^\tau_{i,j}\right\}_{1 \leq i,j \leq n}$ such that

$$M^\tau_{i,j} = \sum_{k=1}^{D} \frac{\mathbb{1}_{(\tau_k \leq i,j < \tau_{k+1})}}{\tau_{k+1} - \tau_k}, \qquad \text{(with } \tau_1 = 1 \text{ and } \tau_{D+1} = n+1 \text{ by convention)}$$

where $\mathbb{1}_{(\tau_k \leq i,j < \tau_{k+1})} = 1$, if $i, j \in [\tau_k, \tau_{k+1}[\cap \mathbb{N}$, and 0 otherwise. Note that $M^\tau_{i,j} \neq 0$ if and only if i, j are in the same segment of τ, which leads to a block-diagonal matrix with D blocks (whose squared Frobenius norm is equal to D). The idea behind the value in each block of this matrix is to define a one-to-one mapping between the set of segmentations in D segments and matrices whose squared Frobenius norm is D.

Let us now consider the matrix $M^{\tau^\star}$ defined from the true segmentation τ^* in the same way. Then, the Frobenius distance between segmentations τ and τ^* is given, through the distance between matrices M^τ and $M^{\tau^\star}$, by

$$d_F\left(\tau, \tau^\star\right) = \left\| M^\tau - M^{\tau^\star} \right\|_F = \sqrt{\sum_{i,j=1}^{n} \left(M^\tau_{i,j} - M^{\tau^\star}_{i,j}\right)^2}.$$

4.4. Results

In our experiments, we successively considered two types of signals: (i) the total copy number profiles (TCN) and (ii) the joint profiles in $\mathbb{R}^2$ made of (TCN,BAF).

4.4.1. Comparison with KS.Lin and ECP for a high tumor percentage (easy case)

First we compare all approaches in the simple case where the tumor percentage is equal to 100%. The performances, using only the TCN or the (TCN,BAF) profiles, are reported in Figure 5 and measured in terms of accuracy (Left) and risk (Right).

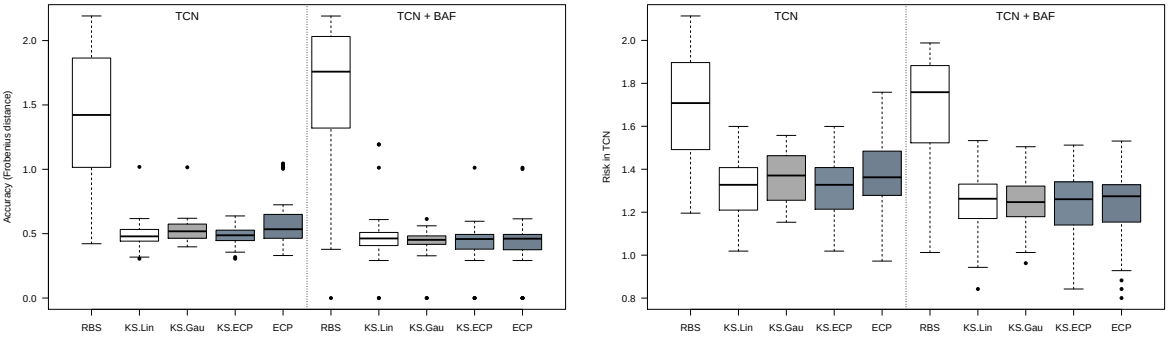

Figure 5: Accuracy (left) and Risk (right) on TCN and (TCN,BAF) profiles with a tumor percentage of 100%. Boxplots of RBS, ECP, KS.Lin, KS.Gau and KS.ECP for their selected number of change-points ($\hat{D}$) are shown.

In all these experiments RBS clearly performs badly. We believe this is mostly due to the poor estimation of the number of segments made by RBS. Indeed the performances of RBS are closer to the ones of other approaches when considering the true number of segments (results not shown here).

We then compare KS.Lin to KS.Gau, KS.ECP, and ECP. With TCN data, KS.Lin has a small advantage over KS.Gau (with an average accuracy difference of 0.03 and a p-value of 0.012) and ECP (with an average accuracy difference of 0.1 and p-values of 0.007). This is also true when considering the risk. KS.ECP has a slightly better empirical average accuracy than KS.Lin but this difference is not significant. Let us also mention that none of the differences are found significant with the (TCN, BAF) profiles. It is our opinion that in this simple scenario all true change-points arise mostly in the mean of the distribution. Thus it is noticeable that the performances of approaches also looking for changes in the whole distribution (like ECP, KS.Gau, KS.ECP) are (almost) on par with those specifically looking for change-points in the mean (like KS.Lin and RBS for the true number of change-points $D^\star$).

We also compared ECP to KS.Gau and KS.ECP. We found no differences except between ECP and KS.ECP for TCN profiles. In that case KS.ECP has significantly better accuracy and risk than ECP. But this difference remains small as can be seen on Figure 5.

Note that for all approaches, performances on (TCN,BAF) profiles are slightly better than those with TCN profiles (p-values smaller than 10^{-4}).

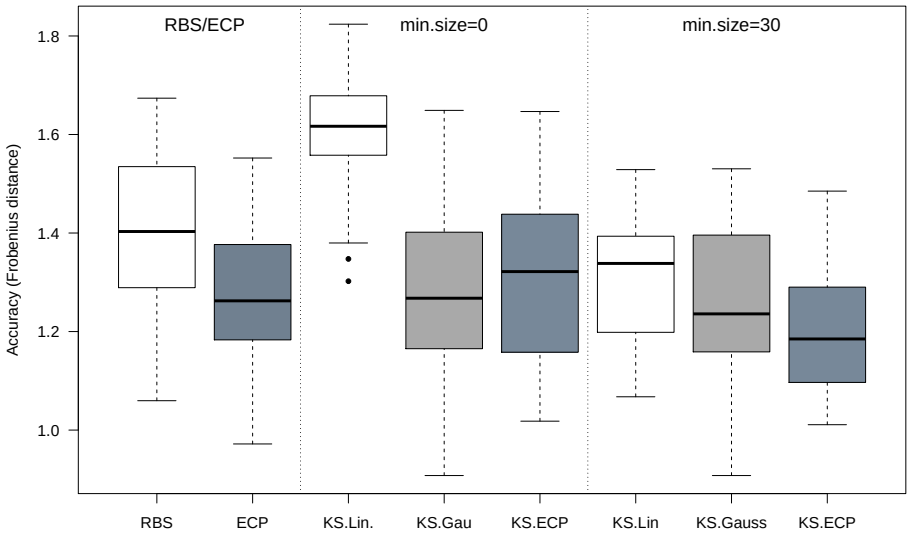

Figure 6: Performances of KS.Lin, KS.Gau and KS.ECP for the true number of changepoints $D^\star = 10$ with or without a constraint on the minimal size of segments (left: $\ell \geq 1$, right: $\ell \geq 30$). The results for RBS and ECP for $D^\star = 10$ are also reported. RBS does not include a constraint while ECP has a default minimal size of 30.

.

4.4.2. Constraint on the segment sizes for a low tumor percentage (difficult case)

We then turn to the more difficult case where the tumor percentage is equal to 50%. In this scenario excluding segments with less than 30 points (as is done by default in ECP) is beneficial. Figure 6 illustrates this strong improvement when adding this constraint to KS.Lin, KS.Gau and KS.ECP and when considering the true number of change-points $D^\star = 10$ (p-values of respectively (8.10^{-9}, 9.10^{-3} and 10^{-4}). More generally it is our experience that such a constraint can greatly improve performances when the signal-to-noise ratio is low. For this reason, in the remainder of our experiments and for a tumor percentage of 50%, we will report results including the constraint on the segment sizes ($\ell = 30$). Let us also mention that for higher tumor percentages adding the constraint does not change the segmentation in $D^\star$ segments recovered by KS.Lin, KS.Gau and KS.ECP.

4.4.3. Comparison with KS.Lin and ECP for a low tumor percentage (difficult case)

We compared KS.Lin to KS.Gau, KS.ECP, and ECP for a tumor percentage of 50%. The accuracy of all these approaches is reported in Figure 7. The minimum length of any segment is fixed at $\ell = 30$ for all approaches (except RBS as it is not possible) and the number of segments is estimated.

In all these experiments RBS performs badly. We believe this is because it poorly selects the number of segments and also because it does not include a constraint on segment sizes.

We compared KS.Lin to KS.Gau, KS.ECP, and ECP. For both TCN and (TCN,BAF) profiles KS.Lin performs worse than KS.Gau, KS.ECP, and ECP in terms of accuracy and risk (all p-

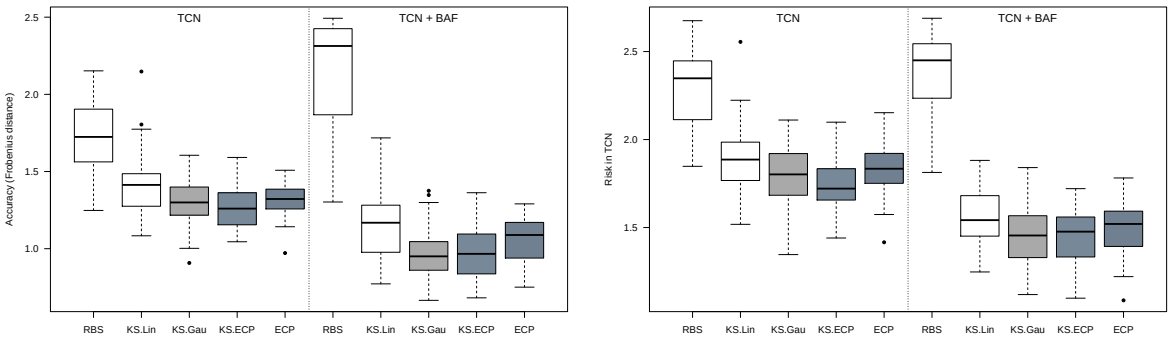

Figure 7: Accuracy (left) and Risk (right) on TCN and (TCN,BAF) profiles with a tumor percentage of 50%. Boxplots of RBS, ECP, KS.Lin, KS.Gau and KS.ECP for their selected number of changepoints ($\hat{D}$) are shown.

values are smaller than 2.10^{-4}). In this more difficult scenario, changes do not arise only in the mean of the distribution, which gives an advantage to approaches looking for changes in the whole distribution and not only in the mean as KS.Lin does.

We then compare ECP to KS.Gau and KS.ECP. First, KS.Gau seems to have a slightly better accuracy and risk than ECP for both TCN and (TCN,BAF) profiles. Two of these differences are found significant with a cut-off of 5% and none with a cut-off of 1%. This leads us to conclude that ECP and KS.Gau have similar performances in the present experiments. Second, KS.ECP has a slightly better accuracy and risk than ECP in TCN for both TCN and (TCN,BAF) profiles. All of these differences are found significant (for the accuracy in (TCN,BAF) $p = 0.0076$, in TCN $p = 0.015$, for the risk in (TCN,BAF) $p = 0.0081$ and in TCN $p = 0.00016$). Although significant these differences remain small (about three times smaller than the differences between KS.Lin and ECP).

Finally it should be noted that KS.ECP is faster than ECP for a profile of $n = 5000$ (5 seconds against 15 minutes). All of this leads to conclude that, in our simulation experiments, KS.Gau and KS.ECP are the best change-point detection procedures among the considered ones since they perform as well as ECP while being by far less memory and time consuming.

4.4.4. Estimation of the number of segments including the minimum length constraint

Let us now assess the behaviour of the model selection procedure derived in Section 2.5 by taking into account the new constraint on the minimal length of the candidate segments.

From Figure 8 it can be seen that whatever the kernel the performances of the *Kernseg* procedure at the estimated number of segments are worse than those at $D^\star$. But this difference remains very small. This empirically validates the use of our modified penalty taking into account a constraint on the size of the segments. We recall that adding this constraint is important in low-signal-to-noise settings, which are common in practice.

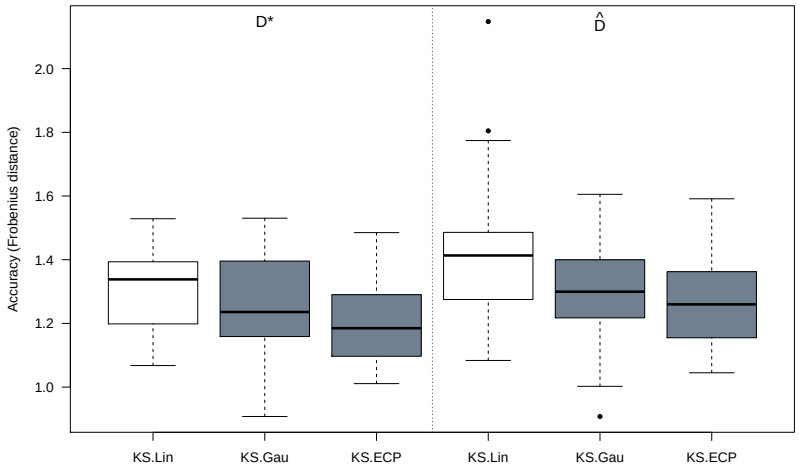

Figure 8: Accuracy of KS.Lin, KS.Gau and KS.ECP on (TCN,BAF) for a tumor percentage of 50% for $D*$ (left) and $\hat{D}$ (right).

4.4.5. Quality of the approximation

The purpose of the present section is to illustrate the behaviour of *ApKS* (in terms of statistical precision) as an alternative to *Kernseg* (which is more time consuming). Since we do not provide any theoretical warranty on the model selection performances of *ApKS*, we only show its results for several values of $p \in \{4, 10, 40, 80, 160\}$ at the true number of segments $D^\star$. For each value of p and each of the TCN and BAF profiles, we compute the approximation by: (i) evaluating the smallest and largest observed value (respectively denoted by m and M), (ii) using an equally spaced grid of p deterministic values between m and M and (iii) use those p values to perform the approximation of the Gram matrix.

From Figure 9 it clearly appears that the number of points used to build the low-rank approximation to the Gram matrix is an influential parameter that has to be carefully fixed. However as long as p is chosen large enough, the approximation seems to provide very similar results. This suggests that one should find a trade-off between the statistical performances and the computation cost. Indeed from a statistical point of view increasing p is beneficial (or at least not detrimental). In contrast from a computationnal point of view increasing p is detrimental and increases the complexity in time ($O(p^2 n)$).

Let us finally emphasize that for large enough p the performances of *ApKS* are very close to those of KS.Gau. Given the low time complexity of *ApKS* compared to KS.Gau we argue that for large profiles ($n \gg 10^5$) *ApKS* could be an interesting alternative to KS.Gau.

Nevertheless several questions related to the use of *ApKS* remain open. For instance, the optimal way to build the low-rank approximation of the Gram matrix is a challenging question which can be embedded in the more general problem of choosing the optimal kernel. Designing a theoretically grounded penalized criterion to perform model selection with *ApKS* is also a crucial problem which remains to be addressed.

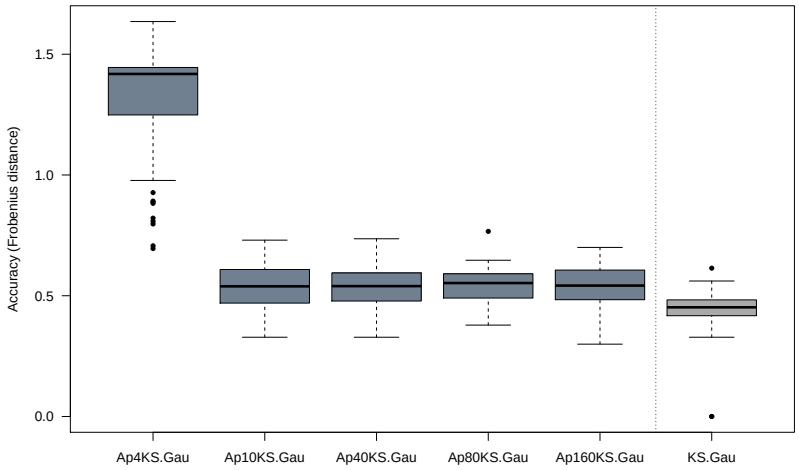

Figure 9: Accuracy of *ApKS* with the Gaussian Kernel and for various p and of KS.Gau on (TCN,BAF) for a tumor percentage of 50% for $D*$.

5. Conclusion

Existing nonparametric change-point detection procedures such as that of [3] exhibit promising statistical performances. Yet their high computational costs (time and memory) are severe limitations that often make it difficult to use for practitioners. Therefore an important task is to develop computationally efficient algorithms (leading to exact or approximate solutions) reducing the time and memory costs of these statistically effective procedures.

In this paper we focus on the multiple change-points detection framework with reproducing kernels. We have detailed a versatile (*i.e.* applicable to any kernel) exact algorithm which is quadratic in time and linear in space. We also provided a versatile approximation algorithm which is linear both in time and space and allows to deal with very large signals ($n \geq 10^6$) on a standard laptop. The computational efficiency in time and space of these two new algorithms has been illustrated on empirical simulation experiments showing that the new algorithms is more efficient than its direct competitor ECP. The statistical accuracy of our kernel-based procedures has been empirically assessed in the setting of DNA copy numbers and allele B fraction profiles. In particular, results illustrate that characteristic kernels (enabling the detection of changes in any moment of the distribution) can lead to better performances than procedures dedicated to detecting changes arising only in the mean.

Acknowledgements

This work has been funded by the French Agence Nationale de la Recherche (ANR) under reference ANR-11-BS01-0010, by the CNRS under the PEPS BeFast, by the CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, and by Chaire d'excellence 2011-2015 Inria/Lille 2.

References

[1] Arlot, S., Celisse, A., 2011. Segmentation of the mean of heteroscedastic data via cross-validation. Stat. Comput. 21 (4), 613–632.

[2] Arlot, S., Celisse, A., Harchaoui, Z., Feb. 2012. Kernel change-point detection. ArXiv:1202.3878v1.

[3] Arlot, S., Celisse, A., Harchaoui, Z., 2012. A kernel multiple change-point algorithm via model selection. arXiv preprint arXiv:1202.3878.

[4] Aronszajn, N., 1950. Theory of reproducing kernels. Transactions of the American mathematical society 68 (3), 337–404.

[5] Aronszajn, N., May 1950. Theory of reproducing kernels. Transactions of the American Mathematical Society 68 (3), 337–404.

[6] Auger, I. E., Lawrence, C. E., 1989. Algorithms for the optimal identification of segment neighborhoods. Bulletin of mathematical biology 51 (1), 39–54.

[7] Auger, I. E., Lawrence, C. E., 1989. Algorithms for the optimal identification of segment neighborhoods. Bulletin of mathematical biology 51 (1), 39–54.

[8] Bach, F., 2013. Sharp analysis of low-rank kernel matrix approximations. In: In Proc. COLT, 2013. pp. 185–209.

[9] Bellman, R., 1961. On the approximation of curves by line segments using dynamic programming. Communications of the ACM 4 (6), 284.

[10] Berg, C., Christensen, J. P. R., Ressel, P., 1984. Harmonic Analysis on Semigroups. Springer, New-York.

[11] Berlinet, A., Thomas-Agnan, C., 2004. Reproducing kernel Hilbert spaces in probability and statistics. Kluwer Academic Publishers, Boston, MA, with a preface by Persi Diaconis.
URL http://dx.doi.org/10.1007/978-1-4419-9096-9

[12] Birgé, L., Massart, P., 2007. Minimal penalties for Gaussian model selection. Probab. Theory Related Fields 138 (1-2), 33–73.

[13] Brodsky, E., Darkhovsky, B. S., 2013. Nonparametric methods in change point problems. Vol. 243. Springer Science & Business Media.

[14] Christmann, A., Steinwart, I., 2010. Universal kernels on non-standard input spaces. In: Advances in neural information processing systems. pp. 406–414.

[15] Cleynen, A., Dudoit, S., Robin, S., 2014. Comparing segmentation methods for genome annotation based on rna-seq data. Journal of Agricultural, Biological, and Environmental Statistics 19 (1), 101–118.

[16] Cleynen, A., Koskas, M., Lebarbier, E., Rigaill, G., Robin, S., 2014. Segmentor3isback: an r package for the fast and exact segmentation of seq-data. Algorithms for Molecular Biology 9, 6.

[17] Cleynen, A., Lebarbier, E., 2014. Segmentation of the poisson and negative binomial rate models: a penalized estimator. ESAIM: Probability and Statistics 18, 750–769.

[18] Cormen, T. H., 2009. Introduction to algorithms. MIT press.

[19] Dieuleveut, A., Bach, F., 2016. Nonparametric stochastic approximation with large step-sizes. The Annals of Statistics 44 (4), 1363–1399.

[20] Drineas, P., Mahoney, M. W., 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research 6, 2153–2175.

[21] Fine, S., Scheinberg, K., Cristianini, N., Shawe-Taylor, J., Williamson, B., 2001. Efficient svm training using low-rank kernel representations. Journal of Machine Learning Research 2, 243–264.

[22] Fryzlewicz, P., 2014. Wild binary segmentation for multiple change-point detection. The Annals of Statistics 42 (6), 2243–2281.

[23] Garreau, D., Arlot, S., 2016. Consistent change-point detection with kernels. arXiv preprint arXiv:1612.04740.

[24] Gartner, T., 2008. Kernels for structured data. Vol. 72. World Scientific.

[25] Gey, S., Lebarbier, E., 2008. Using cart to detect multiple change points in the mean for large sample. Tech. rep., HAL.
URL https://hal.inria.fr/file/index/docid/327146/filename/article_CART.pdf

[26] Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., Sriperumbudur, B. K.,

2012. Optimal kernel choice for large-scale two-sample tests. In: Advances in neural information processing systems. pp. 1205–1213.

[27] Hanahan, D., Weinberg, R. A., 2011. Hallmarks of cancer: the next generation. Cell 144 (5), 646–674.

[28] Harchaoui, Z., Cappé, O., 2007. Retrospective mutiple change-point estimation with kernels. In: Statistical Signal Processing, 2007. SSP'07. IEEE/SP 14th Workshop on. IEEE, pp. 768–772.

[29] Hautaniemi, A. R., Kauraniemi, S., Yli-Harja, P., Astola, O., Wolf, J., Kallioniemi, M., 2003. A cgh-plotter: Matlab toolbox for cgh-data analysis. Bioinformatics 13 (1714–1715).

[30] Haynes, K., Eckley, I. A., Fearnhead, P., 2017. Computationally efficient changepoint detection for a range of penalties. Journal of Computational and Graphical Statistics 26 (1), 134–143.

[31] Hocking, T., Schleiermacher, G., Janoueix-Lerosey, I., Boeva, V., Cappo, J., Delattre, O., Bach, F., Vert, J.-P., 2013. Learning smoothing models of copy number profiles using breakpoint annotations. BMC Bioinformatics 14 (1), 164.

[32] James, N. A., Matteson, D. S., 2013. ecp: An r package for nonparametric multiple change point analysis of multivariate data. arXiv preprint arXiv:1309.3295.
URL http://cran.r-project.org/web/packages/ecp/index.html

[33] Jong, K., Marchiori, E., van der Vaart, A., Ylstra, B., Weiss, M., Meijer, G., 2003. Applications of evolutionary computing. In: EvoWorkshops 2003: Proceedings. Vol. 2611 of chap. chromosomal breakpoint detection in human cancer. Springer-Verlag Heidelberg, pp. 54–65.

[34] Killick, R., Fearnhead, P., Eckley, I., 2012. Optimal detection of changepoints with a linear computational cost. Journal of the American Statistical Association 107 (500), 1590–1598.

[35] Lai, Y., Jun. 2012. Change-Point analysis of paired Allele-Specific copy number variation data. Journal of Computational Biology 19 (6), 679–693.

[36] Lajugie, R., Bach, F., Arlot, S., 2014. Large-margin metric learning for constrained partitioning problems. In: International Conference on Machine Learning. pp. 297–305.

[37] Lebarbier, E., Apr. 2005. Detecting multiple change-points in the mean of gaussian process by model selection. Signal Process. 85 (4), 717–736.
URL http://dx.doi.org/10.1016/j.sigpro.2004.11.012

[38] Ledoux, M., Talagrand, M., 1991. Probability in Banach spaces. Vol. 23 of Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]. Springer-Verlag, Berlin, isoperimetry and processes.

[39] Maidstone, R., Hocking, T., Rigaill, G., Fearnhead, P., 2017. On optimal multiple changepoint algorithms for large data. Statistics and Computing 27 (2), 519–533.

[40] Matteson, D. S., James, N. A., 2014. A nonparametric approach for multiple change point analysis of multivariate data. Journal of the American Statistical Association 109 (505), 334–345.

[41] Matteson, D. S., James, N. A., 2014. A nonparametric approach for multiple change point analysis of multivariate data. Journal of the American Statistical Association 109 (505), 334–345.

[42] Neuvial, P., Bengtsson, H., Speed, T. P., Mar. 2011. Statistical analysis of single nucleotide polymorphism microarrays in cancer studies. In: Handbook of Statistical Bioinformatics, 1st Edition. Springer Handbooks of Computational Statistics. Springer, pp. 225–255.

[43] Olshen, A. B., Venkatraman, E. S., Lucito, R., Wigler, M., 2004. Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics 5 (4), 557–572.

[44] Picard, F., Robin, S., Lavielle, M., Vaisse, C., Daudin, J.-J., Jan. 2005. A statistical approach for array-CGH data analysis. BMC bioinformatics 6, 27.
URL http://www.ncbi.nlm.nih.gov/pubmed/15705208

[45] Pierre-Jean, M., Rigaill, G., Neuvial, P., 2014. Performance evaluation of DNA copy number segmentation methods. Briefings in Bioinformatics.
URL http://bib.oxfordjournals.org/content/early/2014/09/08/bib.bbu026.abstract

[46] Rigaill, G., 2015. 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.

[47] Rigaill, G., Lebarbier, É., Robin, S., 2012. Exact posterior distributions and model selection criteria for multiple change-point detection problems. Statistics and Computing 22 (4), 917–929.

[48] Sejdinovic, D., Sriperumbudur, B., Gretton, A., Fukumizu, K., 2013. Equivalence of distance-based and RKHS-based statistics in hypothesis testing. The Annals of Statistics 41 (5), 2263–2291.

[49] Smola, A. J., Schölkopf, B., 2000. Sparse greedy matrix approximation for machine learning. In: Proceedings of the Seventeenth International Conference on Machine Learning. ICML '00. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 911–918.
URL http://dl.acm.org/citation.cfm?id=645529.657980

[50] Sriperumbudur, B. K., Fukumizu, K., Lanckriet, G. R., 2010. On the relation between universality, characteristic kernels and rkhs embedding of measures. In: Proc. of 13th International Conference on Artificial Intelligence and Statistics. pp. 773–780.

[51] Staaf, J., Lindgren, D., Vallon-Christersson, J., Isaksson, A., et al., Oct. 2008. Segmentation-based detection of allelic imbalance and loss-of-heterozygosity in cancer cells using whole genome SNP arrays. Genome Biol 9 (9), R136.

[52] Williams, C., Seeger, M., 2001. Using the Nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems 13. MIT Press, pp. 682–688.

[53] Yang, T., 2012. Simple binary segmentation frameworks for identifying variation in DNA copy number. BMC bioinformatics 13 (1), 277.