New efficient algorithms for multiple change-point detection with kernels

A. Célisse[e,c], G. Marot[a,c], M. Pierre-Jean[a,b], G.J. Rigaill[1,b]

[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

The present paper focuses on the problem of detecting abrupt changes arising in the full distribution of the observations (not only in the mean or variance). Several statistical approaches based on kernels have been proposed to tackle this problem. Although they enjoy good statistical properties (oracle inequality,...), they suffer a high computational cost (in terms of time and memory) which makes them computationally inefficient even with small to medium sample sizes ($< 10^4$).

Our work addresses this computational issue 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 to deal with medium size signals ($\approx 10^5$).

Secondly, we also design a faster (but approximate) algorithm based on a low-rank approximation to the Gram matrix, which is linear in time and space. This algorithm can be applied to large-scale signals ($n = 10^6$). These exact and approximate 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 our algorithms is observed to be faster than that of other considered procedures. Finally the simulation results empirically confirm the higher statistical accuracy of kernel-based approaches (and their flexibility) to analyze biological profiles made of DNA copy numbers and allele B frequencies.

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

1. Background

1.1. Change-point detection with kernel

In this paper we consider the change-point detection problem [1] 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 [2**?** **?**]. These models allow to detect different types of changes: in the mean, in the variance, and in both the mean and variance (see also [2, 3, 4], ...) 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 in various applications (see for example [5, 6]). 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 [7, 8] is more generic and has the potential to detect any change arising in the full 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 [9]. 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 the dynamic programming algorithm [**?**]. More precisely [7] describe a dynamic programming algorithm to recover the best segmentation from 1 to $D_{\max}$ segments. They claim their algorithm has a $\mathcal{O}(D_{\max} n^2)$ time complexity, but the latter is not described in full details and its straightforward implementation requires the storage of a $n \times n$ cost matrix (personal communication of the first author of [7] who was kind enough to send us his code). The algorithm has a $\mathcal{O}(n^2)$ space complexity, which is a severe limitation in practice with nowadays sample sizes. For instance analyzing a signal of length $n = 10^5$ requires to store a $10^5 \times 10^5$ matrix of doubles, which takes 80 Go. Furthermore computing this cost matrix is not straightforward. In fact simply using formula (8) of [7] 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 [8].

The first one is to describe a new algorithm to simultaneously perform the dynamic programming step of [7] 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 [8] 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 the improvement allowed by our approach can be applied to other existing strategies such as the so-called ECP one [10]. In particular we show that the *divisive clustering algorithm* it is based on (which provides an approximate solution with a complexity of order $O(n^2)$ in time) can be replaced by our algorithm which provides the exact solution with the same time 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, which leads to (almost) the 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 procedure introduced by [8]. 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 [11]. The assessment is done by comparing our approach to state-of-the-art alternatives on resampled real DNA copy number data [12]. This illustrates the versatility of the kernel-based approach allowing to detect changes in the distribution of such complex signals without explicitly modeling the type of change we are looking for.

1.2. Outline of the paper

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 following [8]. The versatility of this kernel-based framework is emphasized by Section 2.5 where it is shown how the ECP approach [10] 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 that of ECP in Section 3.1.3. In Section 3.2 we detail our faster (but approximate) algorithm used to analyze larger signals ($n > 10^5$). It is based on the combination of a low-rank approximation to the Gram matrix and the binary segmentation heuristic [13]. 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 [11].

2. Kernel framework

In this section we recall the framework of [7] where detecting changes in the distribution of a complex signal is rephrased as detecting changes of the mean element of a sequence of point in a Hilbert space. Then we detail the so-called KCP model selection procedure [8], 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* [**?**] throughout the paper. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denote a symmetric positive semi-definite kernel [9], and $\mathcal{H}$ be the associated reproducing kernel Hilbert space (RKHS). We refer to [**?**] 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 interest 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 14, 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 actually 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 [8], 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 [15] that, if $\mathcal{X}$ is separable and $\mathbb{E}\left[\, k(X_t, X_t) \,\right] < +\infty$, then $\mu_t^\star$ exists and is defined as the unique element in $\mathcal{H}$ such that

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

For our purpose we will now exploit the strong connection between $\mu_t^\star$ and $\mathbb{P}_{X_t}$. More precisely for characteristic kernels [16], 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). Exploiting this remark, the idea behind kernel change-point detection [8] is to translate the problem of detecting changes in the distribution into detecting changes in the mean of Hilbert-valued vectors. Let us also notice that the procedure developed by [8] can be interpreted as a "kernelized version" of the procedure proposed by [17], 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 then implies there exists a sequence of $D^* - 1$ true change-points $1 = \tau_0^* < \tau_1^* < \ldots < \tau_{D^*-1}^* < n+1 = \tau_{D^*}^*$ such that

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

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

From a set of $D-1$ candidate change-points $1 < \tau_1 < \ldots < \tau_{D-1} < n+1$, let τ be defined by

$$\tau = (\tau_0, \tau_1, \tau_2, \ldots, \tau_{D-1}, \tau_D),$$

with the convention $\tau_0 = 1$ and $\tau_D = n+1$. With a slight abuse of notation, we also call τ the segmentation of $\{1, \ldots, n\}$ associated with the change-points $1 < \tau_1 < \ldots < \tau_{D-1} < n+1$. Then following [8], the estimator $\hat{\mu}^\tau = (\hat{\mu}_1^\tau, \ldots, \hat{\mu}_n^\tau)' \in \mathcal{H}^n$ of $\mu^* = (\mu_1^*, \ldots, \mu_n^*)'$ is defined by

$$\forall 0 \leq i \leq D-1,\ \forall \tau_i \leq t \leq \tau_{i+1} - 1, \quad \hat{\mu}_t^\tau = \frac{1}{\tau_{i+1} - \tau_i} \sum_{t'=\tau_i}^{\tau_{i+1}-1} Y_{t'}.$$

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

As long as the signal-to-noise ratio remains small, [8] emphasized that all true change-points cannot be recovered without including false change-points. This leadsthem 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, [8] optimize the following penalized criterion

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

where $C \geq 1$ is a constant 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

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

where $\mathcal{T}_D$ denotes the set of segmentations with D segments. This optimization problem, which is usually solved by dynamic programming [**?** **?**], is computationally hard since the cardinality of $\mathcal{T}_D$ is $\binom{n-1}{D-1}$. The second step relies on the straightforward optimization of

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

From a theoretical point of view, this model selection procedure has been proved to be optimal in terms of an oracle inequality by [8], which is the usual non-asymptotic optimality result for model selection procedures [**?**]. However from a practical point of view, the first step (i.e. solving Eq. (7)) remains highly 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 much larger than $n = 10^4$.

2.5. 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 [17], 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 and developed by [10], which 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 $x, y \in \mathbb{R}^d$ and $\|\cdot\|$ denotes the euclidean norm on $\mathbb{R}^d$. Then ρ_α is a semimetric of negative type [18], and for any independent random variables $X, X', Y, Y' \in \mathbb{R}^d$ with respective probability distributions satisfying $P_X = P_{X'}$ and $P_Y = P_{Y'}$, [10] introduce the energy-based distance:

$$\mathcal{E}(X, Y; \alpha) = 2E\left[\,\rho_\alpha(X, Y)\,\right] - E\left[\,\rho_\alpha(X, X')\,\right] - E\left[\,\rho_\alpha(Y, Y')\,\right]. \tag{9}$$

Then following [19] and for every $x_0 \in \mathbb{R}^d$, we define

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

which is a positive semi-definite kernel leading to an RKHS $\mathcal{H}_\alpha^0$. Plugging this in Eq. (9), 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 element of 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 following 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 interesting as our *exact algorithm* (detailed in Section 3.1.2) has a time complexity that is close to that of the *heuristic optimization algorithm* involved in the ECP procedure and called divisive or agglomerative clustering. Therefore for the same computational time, our exact optimization algorithm can replace the approximate one originally used in ECP. Let us also emphasize that our approximate algorithm (exposed in Section 3.2) is even faster.

3. New algorithms

In this section we first show how to avoid the preliminary calculation of the cost matrix required by [7] 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 [7]. This leads to our new exact algorithm (Algorithm 3), which has a reduced space complexity of order $\mathcal{O}(D_{\max} n)$ compared to $\mathcal{O}(n^2)$ for the one used in [7].

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

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

As exposed in Section 2.4, the main computational cost results from Eq. (7), which consists in recovering the best segmentation with $1 \leq D \leq D_{\max}$ segments, that is solving

$$\begin{aligned} \mathbf{L}_{D,n+1} &= \min_{\tau \in \mathcal{T}_D} \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} && \text{(best fit to the data)} \\ \hat{m}_D &= \arg\min_{\tau \in \mathcal{T}_D} \|Y - \hat{\mu}^\tau\|^2_{\mathcal{H},n} && \text{(best segmentation)} \end{aligned} \tag{10}$$

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 [20, 21], which provides the exact solution to the optimization problem (10). 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 $\{\tau_d, ..., \tau_{d+1} - 1\}$ of the segmentation τ has a cost that is equal to

$$C_{\tau_d, \tau_{d+1}} = \sum_{i=\tau_d}^{\tau_{d+1}-1} k(X_i, X_i) \; - \; \frac{1}{\tau_{d+1} - \tau_d} \sum_{i=\tau_d}^{\tau_{d+1}-1} \sum_{j=\tau_d}^{\tau_{d+1}-1} k(X_i, X_j). \tag{11}$$

Then the cost of the segmentation τ is given by

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

which is clearly *segment additive* [7, 8].

Dynamic programming solves (10) for all $1 \leq D \leq D_{\max}$ by applying the following update rules

$$\forall 2 \leq D \leq D_{\max}, \quad \mathbf{L}_{D,n+1} = \min_{\tau \leq n} \{ \; \mathbf{L}_{D-1,\tau} + C_{\tau,n+1} \; \}, \tag{12}$$

which exploits the property that the optimal segmentation in D segments over $\{1, \dots, n\}$ can be computed from optimal ones with $D-1$ segments over $\{1, \dots, \tau\}$ ($\tau \leq n$). Furthermore making the key assumption that *the cost matrix* $\{C_{i,j}\}_{1 \leq i,j \leq n+1}$ *has been stored*, we can compute $\mathbf{L}_{D,n+1}$ following 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, which is the one used by [7], suffers two main limitations. First it assumes that the $C_{\tau,\tau'}$ have been already computed, and does not take into account the resulting computational cost. 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 $O(\tau')$ 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. (11)). Therefore there is no hope to solve (10) exactly in less than quadratic time without additional assumptions on the kernel.

From Eq. (11) let us also remark that computing each $C_{i,j}$ ($1 \leq i < j \leq n$) naively requires itself a quadratic number of operations, hence a $\mathcal{O}(n^4)$ time complexity for computing the whole cost matrix. Then the dynamic programming step (Line 3 of Algorithm 1) is not the limiting factor in that case and the overall time complexity of Agorithm 1 is $\mathcal{O}(D_{\max} n^4)$.

Finally let us also 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 positive semi-definite kernels and the associated Gram matrix [22].

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 Algorithm 1.

Algorithm 2 Improved space complexity

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

Importantly this swap does not change the output of the algorithm and does not induce any additional calculations. Furthemore 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 $\mathcal{O}(n)$. Storing only this column leads to a much improved $\mathcal{O}(D_{\max} n)$ space complexity.

Algorithm 2 finally requires to store 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 whole column.

Iterative computation of the columns of the cost matrix. The last ingredient of our final 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. (11) 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 $k(x_i, x_j)$ is in $\mathcal{O}(1)$, updating from τ' to $\tau' + 1$ requires $\mathcal{O}\left(\tau'\right)$ operations. Note that for many classical kernels, computing $k(x_i, x_j)$ is indeed in $\mathcal{O}(1)$. For example if $x_i \in \mathbb{R}^q$ with q a constant larger or equal to 1 and $k(\cdot, \cdot)$ denotes the Gaussian kernel, each evaluation of $k(x_i, x_j)$ has a $\mathcal{O}(q) = \mathcal{O}(1)$ time complexity. If q is not negligible,

the last example illustrates that the resulting time complexity is only increased by a multiplicative factor.

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

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 [7].

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 $D_{\max} = 40$.

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

Second, we also compare the runtime of Algorithm 3 with that of ECP discussed in Section 2.5 implemented in the R-package [23] (see the left panel of Figure 1). Since ECP is based on the binary segmentation heuristic [24] 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 Algorithm 3. 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 compare our approach and ECP with and without the permutation layer. Finally it is also necessary to emphasize that unlike Algorithm 3, ECP does not provide the exact but only an approximate solution to the optimization problem in Eq. (10). Results are summarized in Figure 1 (Right). It illustrates that our exact algorithm (Algorithm 3) has a quadratic complexity similar to that

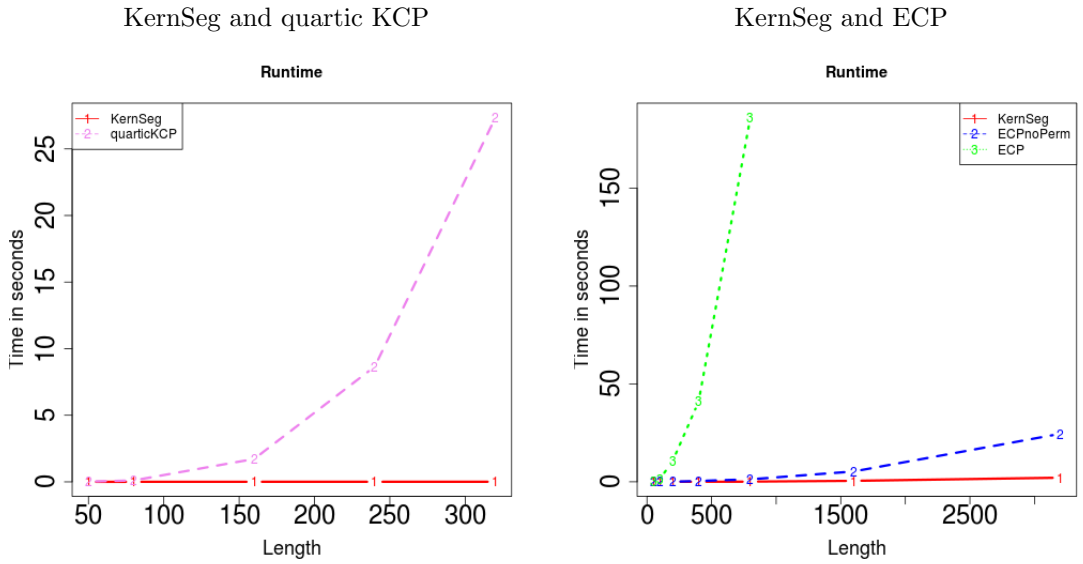

Figure 1: (Left) Runtime for $n < 250$ of Algorithm 1 and Algorithm 3 as a function of the length of the signal (n) for $D_{\max} = 40$. (1-red) and a quartic computation of the cost matrix (2-violet). (Right) Runtime for $n < 2500$ of Algorithm 3 as a function of the length of the signal (1-red) and of ECP without permutation (2-blue) and ECP with the default number of permutations (3-green).

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 Algorithm 3 is exact unlike ECP. Let us also mention that for $n = 10^5$ (with $D_{\max} = 40$), our exact approach runs in 40 minutes (see also Figure 2), whereas ECP (without permutation) would require 7 hours (time estimated from Figure 2 by assuming a quadratic dependency with respect to n).

3.2. Approximating the Gram matrix to speed up the algorithm

In Section 3.1.2, we described an improved algorithm based on carefully combining dynamic programming and the computation of the cost matrix elements. This new algorithm (Algorithm 3) provides the exact solution to the optimization problem given by Eq. (10). However without any further assumption on the underlying kernel, this algorithm only achieves the complexity $\mathcal{O}(n^2)$ in time, which is a clear limitation with large scale signals ($n > 10^6$). Note also that this limitation results from the use of 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 [24], which is known to be computationally efficient for parametric models, suffers the same $\mathcal{O}(n^2)$ time complexity when used in the kernel framework (see Section 3.2.2).

Let us remark however that for some 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 trick

$$\sum_{1 \leq i \neq j \leq n} k(X_i, X_j) = \sum_{1 \leq i \neq j \leq n} \langle X_i, X_j \rangle_{\mathbb{R}^d} = \sum_{1 \leq i \leq n} \left\langle X_i, \sum_{j=1}^{n} X_j - X_i \right\rangle \tag{13}$$
$$= \left\| \sum_{i=1}^{n} X_i \right\|^2 - \sum_{i=1}^{n} \|X_i\|^2 ,$$

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

The purpose of the present section is to describe a general strategy (i.e applicable to any kernel) relying on a low-rank approximation to the Gram matrix [25, 26, 27]. This approximation allows to considerably reduce the computation time by exploiting (13). Note however that the resulting procedure achieves this lower time complexity at the price of only providing an approximation to the exact solution to (10) (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 [28] 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, [28] 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 $\mathbf{K}$ [28, 22] by

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

where $I_n = \{1, \ldots, n\}$, 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

$$\widetilde{\mathbf{K}}_{i,j} = Z_i' Z_j. \tag{14}$$

The main interest of this approximation is that, using Eq. (13), 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.

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 [25, 28, 22] where this point has been extensively discussed.

3.2.2. Binary segmentation heuristic

Since the low-rank approximation of the Gram matrix detailed in Section 3.2.1 leads to deal with finite dimensional vectors in $\mathbb{R}^p$ (14), 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. (10) for each $1 \leq D \leq D_{\max}$, and (ii) to deal with large sample sizes ($n > 10^6$).

The first algorithm is the usual version of constrained dynamic programming [21]. Although it has been recently revisited by [29, 30], this algorithm suffers a worst-case complexity of $\mathcal{O}(n^2)$ in time, which excludes dealing with large sample sizes. In the same line, another version of regularized dynamic programming has been explored by [31] 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. On the one hand, with PELT it is not straightforward to efficiently solve Eq. (10) for each $1 \leq D \leq D_{\max}$, which is precisely the goal we pursue. On the other hand the complexity of the pruning inside PELT depends on the true number of change-points and number of dimensions. In particular for a small number of change-points it is quadratic.

A second possible algorithm is the so-called *binary segmentation* [32, 13, 24] that is a standard heuristic for approximately solving Eq. (10) 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. (10) for

each $1 \leq D \leq D_{\max}$. Furthermore if computing the cost of any segment is linear in its length, then the time complexity of binary segmentation for splitting one segment of length t into two new ones is $\mathcal{O}(t)$. Therefore the overall time complexity of binary segmentation for recovering approximate solutions to (10) for all $1 \leq D \leq D_{\max}$ is only $\mathcal{O}\left(\log(D_{\max})n\right)$ on average and $\mathcal{O}\left(D_{\max}n\right)$ at worse.

An important remark is that binary segmentation only achieves this reduced time complexity provided computing the cost of one segment has a complexity linear in its length. This is precisely what has been allowed by the low-rank matrix approximation summarized by Eq. (14). Otherwise with a quadratic complexity for computing the cost of one segment, binary segmentation would suffer an overall time complexity of order $\mathcal{O}(D_{\max}n^2)$.

3.2.3. Implementation and runtimes of the approximate solution

The approximate 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 worst-case time complexity is then $\mathcal{O}(p^2n + D_{\max}n)$, which allows to deal with large sample sizes ($n > 10^6$).

From this time complexity it arises that an influential parameter is the number p of columns of the Gram 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 potted the runtime of the exact algorithm (Algorithm 3) detailed in Section 3.1.2.

Our fast approximate algorithm 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 in practice with large signals. Note also that Figure 2 illustrates our approximate algorithm returns the solution in a matter of seconds with a sample size of $n = 10^5$, which is much faster than the exact algorithm (based on dynamic programming) that requires a few minutes.

4. Segmentation assessment

From a statistical point of view our exact algorithm provides the same results as that of [8]. However it greatly improves on the latter in terms of computational complexity as proved in Section 3.1. In their simulation experiments [8] 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 positive semi-definite kernels to these biological data: (i) kernels avoid the need for modeling

KernSeg Exact and Heuristic

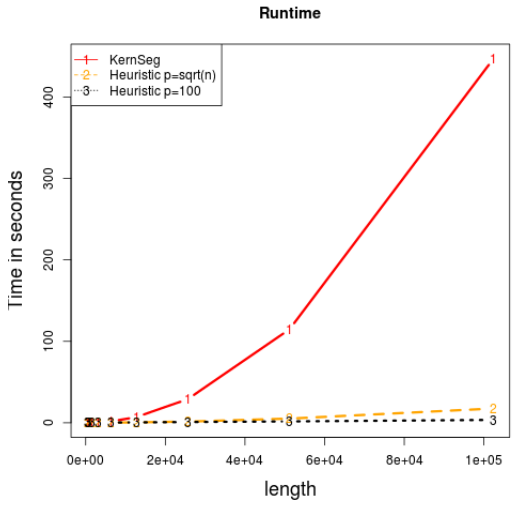

Figure 2: Runtime for $n < 10^5$ of our exact algorithm (Algorithm 3) as a function of n (length of the signal) for $D_{max} = 10$. (1-red), of our approximate algorithm with $p = \sqrt{n}$ (2-orange) and $p = 100$ (3-black).

.

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, allowing 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 [33]. 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 [34]. The top panel (denoted by c) represents estimates of the total copy number (TCN). The bottom panel

(denoted by b) represents estimates of allele B fractions (BAF). We refer to [34] for an explanation of how these estimates are obtained. In the normal region [0-2200], the TCN is centered around two copies and BAF has three modes at 0, 1/2 and 1. Importantly any change in only one of the parental

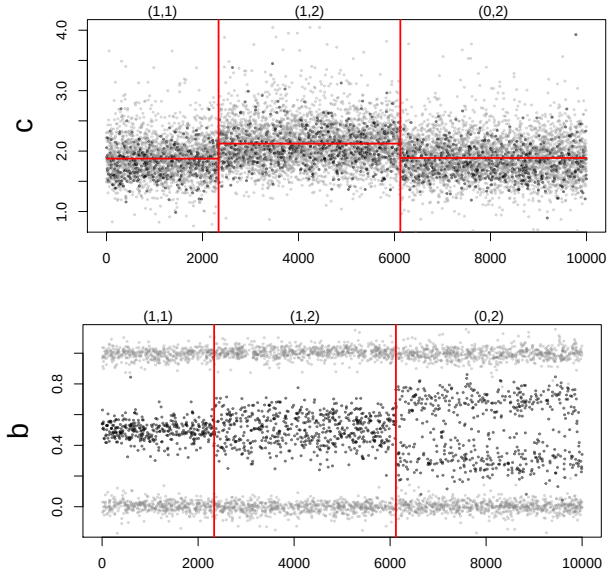

Figure 3: SNP array data. Total copy numbers (c), allelic ratios (b) along 10,000 genomic loci. Red vertical lines represent change-points, and red horizontal lines represent estimated mean signal levels between two change-points. SNPs that are heterozygous in the germline are colored in black; all others are colored in gray.

copy numbers is reflected in both c and b for heterozygous SNPs. Therefore it makes sense to jointly analyze both dimensions to ease the identification of change-points.

4.1.2. Generated data

Realistic DNA profiles with known truth (similar to that of Figure 3) have been generated using the `acnr` package [12]. The constituted benchmark consists of profiles with $5,000$ positions of heterozygous SNPs and exactly $K = 10$ change-points. As in [12] we impose the constraint that segments are either normal, copy-neutral LOH[1], single copy-gain or hemizygous deletion. The `acnr` package allows to vary the difficulty level by adding normal cell contamination, thus degrading tumor purity. Three levels of difficulty have been considered by varying tumor purity with proportions 100% (easy case), 70%, and 50% (difficult case). Figure 4 displays three examples of simulated profiles (one for each tumor purity level).

For each level, $N = 200$ profiles (with the same segment states and change-points) are generated making a total of 600 simulated profiles both for BAF and TCN.

[1] A region where one of parental chromosome has been duplicated and the other one deleted

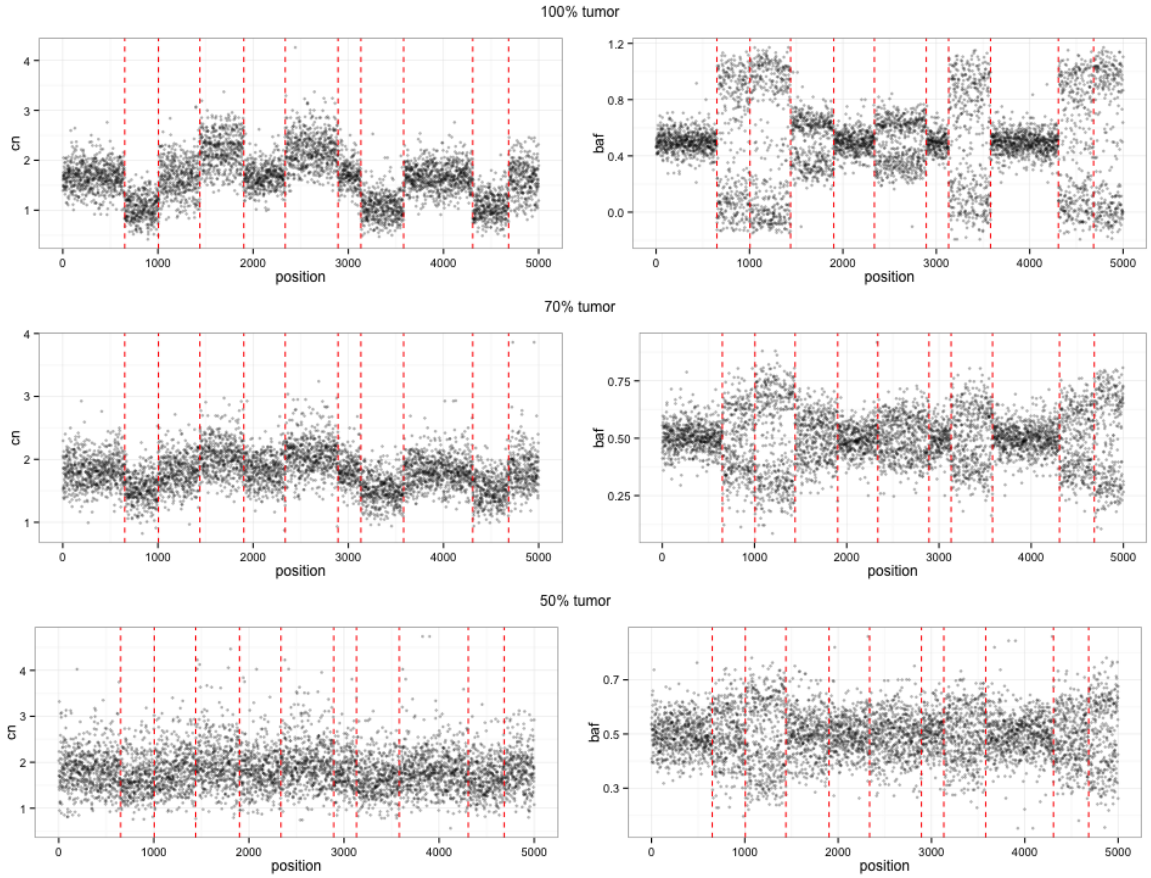

Figure 4: Benchmark1: Simulated profiles with `acnr` package. Each line corresponds to a tumor purity level (100%, 70% and 50%) and column to the copy number (c) and the allele B fraction (b)

4.2. Competing procedures

4.2.1. Kernseg and approximate Kernseg

The implemented algorithm *Kernseg* (corresponding to Algorithm 3) and its fast (but approximate) version based on binary segmentation, denoted by *ApproxKernseg* , remain applicable with any kernel (Gaussian, exponential, polynomial,...). In our simulation experiments we consider two kernels. The first one is the so-called linear kernel defined by $k(x, y) = \langle x, y \rangle$, where $x, yin\mathbb{R}$. It is used as a benchmark since *Kernseg* with this kernel reduces to the procedure of [17]. 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, it is a natural choice to detect any abrupt changes arising in the full distribution [8].

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 a mixture of two coordinate-wise Gaussian kernels, that is

$$k(x_1, x_2) = k_\delta(c_1, c_2) + k_{\delta'}(b_1, b_2) \tag{15}$$

with $x_1 = (c_1, b_1)$ and $x_2 = (c_2, b_2)$ where the first coordinate refers to TCN and the second one to BAF.

The Gaussian kernel is used with $\delta = 1$ with TCN profiles and $\delta' = 0.02$ with BAF ones. Note that several values of δ and δ' have been explored. These ones have been selected as they provide the most representative results.

4.2.2. Recursive Binary Segmentation (RBS)

In the quite recent paper by [12], it has been shown that the Recursive Binary Segmentation (RBS) [35] is the overall state-of-the-art change-point procedure in the present biological context of TCN and BAF profiles.

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 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 by use of 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 as follows. First, for any segmentation $\tau = (\tau_1, \tau_2, \ldots, \tau_D, \tau_{D+1})$, let us introduce a matrix $M = \{M_{i,j}\}_{1 \leq i,j \leq n}$ such that

$$M_{i,j} = \sum_{k=1}^{D+1} \frac{\mathbb{1}_{(\tau_{k-1} \leq i,j < \tau_k)}}{\tau_k - \tau_{k-1}},$$

where $\mathbb{1}_{(\tau_{k-1} \leq i,j < \tau_k)} = 1$, if $i, j \in [\tau_{k-1}, \tau_k[\cap \mathbb{N}$, and 0 otherwise. Let us now consider the matrix M^* 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 and M^*, by

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

4.4. Results

In our experiments, we successively considered three types of signals: (i) total copy number profiles (TCN), (ii) allele B fraction profiles (BAF), and (iii) the joint signal in $\mathbb{R}^2$ made of (TCN,BAF). In the following, results are reported only for the first and the last one since TCN and BAF provide similar results to each other in the present simulation setting.

We first compare our procedure called *Kernseg* to the state-of-the-art segmentation techniques which are usually applied on TCN profiles only. Second, the best competitor is kept and compared to *Kernseg* on the joint signal (TCN, BAF). Lastly, results of *ApproxKernseg* are compared to the ones of *Kernseg* .

4.4.1. Comparison with state-of-the-art procedures on the total copy number (TCN) profiles

In this first set of simulation experiments, two procedures using *Kernseg* (one with the linear kernel denoted by KSegLin, and the other with the Gaussian kernel denoted by KSegGauss) and the RBS approach are run. Figure 5 illustrates the performance of segmentation procedures both in terms of change-points location and in terms of risk. In both cases, the first three boxplots correspond to evaluations performed at the true number of change-points, while the last one illustrates the performance of *Kernseg* (with the Gaussian kernel), including the choice of the number of change-points which has been made following the procedure described by [8].

Using kernels improves the performance of change-point detection. Running *Kernseg* both with Gaussian and linear kernels illustrates the interest of considering non-linear kernels. When used with the linear kernel, our procedure exactly reduces to what Lebarbier (2005) is doing, which is known to work well for detecting changes arising in the mean of a signal. When applying these procedures on TCN only, we observe that linear and Gaussian kernels exhibit a similar good behavior when changes arise

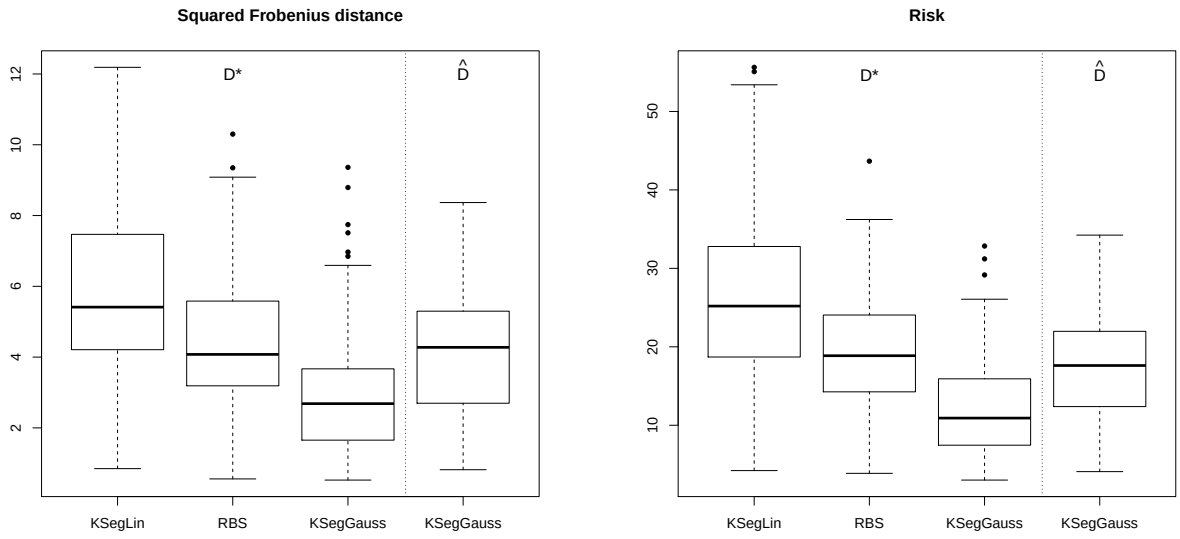

Figure 5: Performances of segmentations on 200 simulated TCN profiles in a difficult setting (tumor purity around 50%)

in the mean and the purity is high (results not shown). This is no longer the case as the purity is low (50%), as displayed in Figure 5. Whatever the chosen performance measure (risk or Frobenius distance), the linear kernel clearly fails in this difficult situation whereas the Gaussian one still provides accurate change-points estimates. Also, this does not result from the choice of the number of segments since the performance of the Gaussian kernel at the selected number of segments remains better than that of the linear kernel at the true one. These results emphasize the advantage provided by non-linear kernels. Using the Gaussian kernel allows to detect changes also arising in higher moments of the signal than the mean. This is noticeable from the risk of the resulting estimator which is lower than that of the linear kernel while the latter precisely focuses on minimizing this risk.

Another important aspect is that the Gaussian kernel performs well both in terms of change-point locations (see Squared Frobenius distance) and in terms of risk. As both criteria lead to similar conclusions in our simulation settings, only figures showing the performance in terms of Squared Frobenius Distance will be displayed in the following. From now on, *Kernseg* will be run with the Gaussian kernel.

KernSeg with Gaussian kernel provides better results than RBS. As mentioned earlier in Section 4.2.2, the Recursive Binary Segmentation (RBS) is the overall state-of-the-art change-point procedure in the present context. When considering the true number of change-points, *Kernseg* with Gaussian kernel provides a lower risk than RBS (see Figure 5) and more accurate change-point locations.

Figure 5 also illustrates that the procedure selecting by itself the number of change-points after the use of *Kernseg* with the Gaussian kernel provides similar results to those of RBS at the true

dimension. This highlights that the sequence of candidate change-points provided by *Kernseg* based on the Gaussian kernel is of higher quality than that of RBS (at least in the considered settings).

4.4.2. Comparison of Kernseg *with RBS on the joint (TCN, BAF) signal*

An important question is to know whether performing a joint segmentation with this type of biological data would improve the performance.

Figure 6 illustrates the gain resulting from the joint segmentation. It corresponds to the difficult situation where tumor purity is around 50%. To ease the comparison, the first three boxplots are the same as those of Figure 5, when applying RBS and *Kernseg* with Gaussian on TCN only. Alongside these boxplots, results for the joint signal are plotted for the same three procedures: RBS and *Kernseg* at the true number of change-points, and *Kernseg* at the estimated number of change-points.

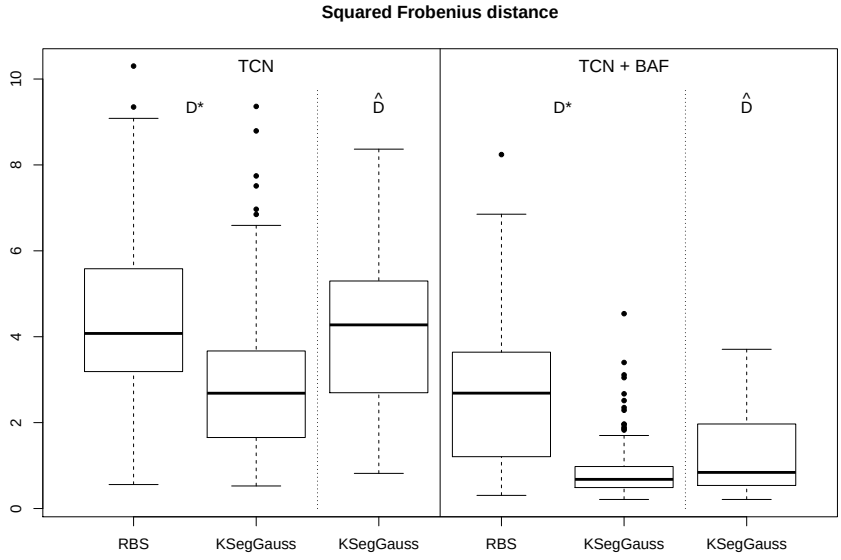

Figure 6: Joint (TCN+ BAF) segmentation performances (right) compared to TCN segmentation performances (left) on 200 simulations in a difficult setting (tumor purity around 50%)

Joint (TCN, BAF) segmentation is better than one-dimensional segmentation. For all procedures, we observe that the risk of the TCN segmentation is higher than the one of the joint signal corresponding segmentation. This supports the idea that TCN and BAF both carry a somewhat complementary piece of information.

Using kernels offers simplicity. An important byproduct of the kernel machinery is that it is easy to make data fusion, which consists here in doing the joint segmentation of the signal, for instance by simply mixing coordinate-wise kernels in the same way as in Eq. (15).

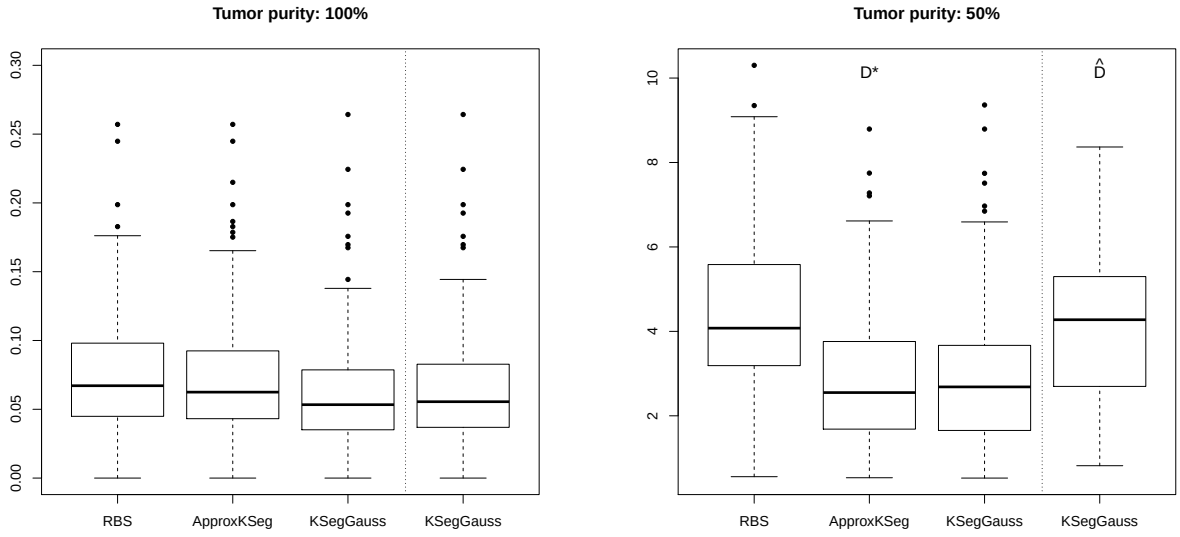

Figure 7: Performance of segmentations using the squared Frobenius Distance. Simulations of 200 TCN profiles. Simple setting with 100% purity (left) and difficult setting with 50% purity (right).

KernSeg with Gaussian kernel outperforms RBS. When considering the true number of change-points in TCN profiles, it was already shown that *Kernseg* with Gaussian kernels was outperforming RBS. When looking at the joint segmentation, the results are even more striking. Indeed, even when the number of segments is estimated, *Kernseg* outperforms RBS at the true number of change-points.

4.4.3. Heuristic Kernseg *with Gaussian kernel*

The approximate algorithm *ApproxKernseg* is motivated by the need for a change-point detection procedure with linear-time complexity. It is derived from *Kernseg* that is a theoretically grounded procedure. However as a heuristic it is not fully theoretically justified. It is out of the scope of the present work to justify it. As a consequence, we do not provide any data-driven choice of the number of segments for *ApproxKernseg* . The purpose of the present section is to empirically illustrate how accurate *ApproxKernseg* can be in terms of statistical accuracy (measured in terms of Frobenius distance) while reducing the computation time.

Figure 7 shows the results obtained either in a simple setting (high tumor purity) or a difficult setting (low tumor purity) on TCN profiles. Scales are different between these two settings. In particular, differences are clearly greater in the second setting compared to the first one. In the simple setting, all procedures perform equally well and present median values for squared Frobenius Distances between 0.05 and 0.07. In the difficult setting, the heuristic performs as well as *Kernseg* at the true number of change-points and still better than RBS.

5. Conclusion

Existing nonparametric change-point detection procedures such as that of [8] exhibit promising statistical performances. Yet their high computational costs (time and memory) are strong limitations that often make it difficult for practitioners to use them. Therefore an important task is to develop computationally efficient (exact or approximate) algorithms allowing to reduce the time and memory costs of these statistically effective procedures.

In this paper we focus on kernel change-point detection framework. We have detailed a generic (*i.e.* applying to any kernel) exact algorithm which is quadratic in time and linear in space. We also provided a generic approximate algorithm which is linear both in time and space and allows to deal with huge signals ($n \geq 10^6$) on a standard laptop. The computational efficiency of these two new algorithms has been illustrated on empirical simulation experiments, where it arises that the new algorithms outperforms other ongoing nonparametric procedures. The statistical accuracy of our procedures has been empirically assessed in the particular 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 mainly 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] E. G. Carlstein, H.-G. Müller, D. Siegmund, Change-point problems, IMS, 1994.

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

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

[4] F. Picard, S. Robin, M. Lavielle, C. Vaisse, J.-J. Daudin, A statistical approach for array-CGH data analysis., BMC bioinformatics 6 (2005) 27. doi:10.1186/1471-2105-6-27.
URL http://www.ncbi.nlm.nih.gov/pubmed/15705208

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

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

[7] Z. Harchaoui, O. Cappé, Retrospective multiple change-point estimation with kernels, in: IEEE Workshop on Statistical Signal Processing, 2007.

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

[9] N. Aronszajn, Theory of reproducing kernels.

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

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

[12] M. Pierre-Jean, G. Rigaill, P. Neuvial, Performance evaluation of DNA copy number segmentation methods, Briefings in BioinformaticsarXiv:http://bib.oxfordjournals.org/content/early/2014/09/08/bib.bbu026.full.pdf+html, doi:10.1093/bib/bbu026.
URL http://bib.oxfordjournals.org/content/early/2014/09/08/bib.bbu026.abstract

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

[14] T. Gřtner, Kernels for structured data, Vol. 72 of Series in Machine Perception and Artificial Intelligence, World Scientific Publishing, 2008.

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

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

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

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

[19] D. Sejdinovic, B. Sriperumbudur, A. Gretton, K. Fukumizu, Equivalence of equivalence and distance-based and rkhs-based statistics in hypothesis testing, The Annals of Statistics.

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

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

[22] F. Bach, Sharp analysis of low-rank kernel matrix approximations., in: In Proc. COLT, 2013, 2013.

[23] N. A. James, D. S. Matteson. [link].
URL http://cran.r-project.org/web/packages/ecp/index.html

[24] P. Fryzlewicz, Wild Binary Segmentation for Multiple Change-Point Detection, Annals of Statistics, to appear.

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

[26] A. J. Smola, B. Schökopf, 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, 2000, pp. 911–918.
URL http://dl.acm.org/citation.cfm?id=645529.657980

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

[28] P. Drineas, M. W. Mahoney, On the nyström method for approximating a gram matrix for improved kernel-based learning, JMLR 6 (2153–2175).

[29] G. Rigaill, Pruned dynamic programming for optimal multiple change-point detection, Tech. rep., http://arXiv.org/abs/1004.0887 (2010).

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

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

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

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

[34] P. Neuvial, H. Bengtsson, T. P. Speed, Statistical analysis of single nucleotide polymorphism microarrays in cancer studies, in: Handbook of Statistical Bioinformatics, 1st Edition, Springer Handbooks of Computational Statistics, Springer, 2011.

[35] S. Gey, E. Lebarbier, Using CART to detect multiple change points in the mean for large sample, Tech. rep., Statistics for Systems Biology research group (2008).