Nested Kriging predictions for datasets with large number of observations

Didier Rullière[*], Nicolas Durrande[†], François Bachoc[‡] and Clément Chevalier[§]

Thursday 20th July, 2017

Abstract

This work falls within the context of predicting the value of a real function at some input locations given a limited number of observations of this function. The Kriging interpolation technique (or Gaussian process regression) is often considered to tackle such a problem but the method suffers from its computational burden when the number of observation points is large. We introduce in this article nested Kriging predictors which are constructed by aggregating sub-models based on subsets of observation points. This approach is proven to have better theoretical properties than other aggregation methods that can be found in the literature. Contrarily to some other methods it can be shown that the proposed aggregation method is consistent. Finally, the practical interest of the proposed method is illustrated on simulated datasets and on an industrial test case with 10^4 observations in a 6-dimensional space.

Keywords: Gaussian process regression · big data · aggregation methods · best linear unbiased predictor · spatial processes.

1 Introduction

Gaussian process regression models have proven to be of great interest in many fields when it comes to predict the output of a function $f : D \to \mathbb{R}$, $D \subset \mathbb{R}^d$, based on the knowledge of n input-output tuples $(x_i, f(x_i))$ for $1 \leq i \leq n$ [Stein, 2012, Santner et al., 2013, Williams and Rasmussen, 2006]. One asset of this method is to provide not only a mean predictor but also a quantification of the model uncertainty. The Gaussian process regression framework uses a (centered) real-valued Gaussian process Y over D as a prior distribution for f and approximates it by the conditional distribution of Y given the observations $Y(x_i) = f(x_i)$ for $1 \leq i \leq n$. In this framework, we denote by $k : D \times D \to \mathbb{R}$ the covariance function (or kernel) of Y: $k(x, x') = \mathrm{Cov}\left[Y(x), Y(x')\right]$, and by $X \in D^n$ the vector of observation points with entries x_i for $1 \leq i \leq n$.

In the following, we use classical vectorial notations: for any functions $f : D \to \mathbb{R}$, $g : D \times D \to \mathbb{R}$ and for any vectors $A = (a_1, \ldots, a_n) \in D^n$ and $B = (b_1, \ldots, b_m) \in D^m$, we denote by $f(A)$ the $n \times 1$ real valued vector with components $f(a_i)$ and by $g(A, B)$ the $n \times m$ real valued matrix with components $g(a_i, b_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. With such notations, the conditional distribution of Y given the $n \times 1$ vector of observations $Y(X)$ is Gaussian with mean, covariance and variance:

$$\begin{cases} M_{full}(x) = \mathrm{E}\left[Y(x)|Y(X)\right] = k(x, X)k(X, X)^{-1}Y(X)\,, \\ c_{full}(x, x') = \mathrm{Cov}\left[Y(x), Y(x')|Y(X)\right] = k(x, x') - k(x, X)k(X, X)^{-1}k(X, x')\,, \\ v_{full}(x) = c_{full}(x, x)\,. \end{cases} \tag{1}$$

[*]Université de Lyon, Université Claude Bernard Lyon 1, ISFA, Laboratoire SAF, EA2429, 50 avenue Tony Garnier, 69366 Lyon, France, didier.rulliere@univ-lyon1.fr.

[†]*Corresponding author*, Mines Saint-Étienne, H. Fayol Institute, 158 cours Fauriel, Saint-Étienne, France and CNRS LIMOS, UMR 5168, durrande@emse.fr.

[‡]Institut de Mathématiques de Toulouse, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse, France, francois.bachoc@math.univ-toulouse.fr.

[§]Institute of Statistics, University of Neuchâtel, avenue de Bellevaux 51, 2000 Neuchâtel, Switzerland, clement.chevalier@unine.ch.

Since we do not specify yet the values taken by Y at X, the "mean predictor" $M_{full}(x)$ is random so it is denoted by an upper-case letter M. The approximation of $f(x)$ given the observations $f(X)$ is thus given by $m_{full}(x) = \mathrm{E}\left[Y(x)|Y(X) = f(X)\right] = k(x, X)k(X, X)^{-1}f(X)$. This method is quite general since an appropriate choice of the kernel allows to recover the models obtained from various frameworks such as linear regression and splines models [Wahba, 1990].

One limitation of such models is the computational time required for building models based on a large number of observations. Indeed, these models require computing and inverting the $n \times n$ covariance matrix $k(X, X)$ between the observed values $Y(X)$, which leads to a $O(n^2)$ complexity in space and $O(n^3)$ in time. In practice, this computational burden makes Gaussian process regression difficult to use when the number of observation points is in the range $[10^3, 10^4]$ or greater.

Many methods have been proposed in the literature to overcome this limit. Let us first mention that, when the observations are recorded on a regular grid, choosing a separable covariance function k enables to drastically simplify the inversion of the covariance matrix $k(X, X)$, since the latter can be written as a Kronecker product. In the same context of gridded data, alternative approaches such as Gaussian Markov Random Fields are also available [Rue and Held, 2005].

For irregularly spaced data, a common approach in machine learning relies on inducing points. It consists in introducing a set W of pseudo input points and in approximating the full conditional distribution $Y(x)|Y(X)$ by $Y(x)|Y(W)$. The challenge here is to find the best locations for the inducing inputs and to decide which values should be assigned to the outputs at W. Various methods are suggested in the literature to answer these questions [Guhaniyogi et al., 2011, Hensman et al., 2013, Katzfuss, 2013, Zhang et al., 2015]. One drawback of this kind of approximation is that the predictions do not interpolate the observation points any more. Note that this method has recently been combined with the Kronecker product method in [Nickson et al., 2015].

Other methods rely on low rank approximations or compactly supported covariance functions. Both methods show limitations when the scale dependence is respectively short and large. For more details and references, see [Stein, 2014, Maurya, 2016, Bachoc et al., 2017]. Another drawback – which to the best of our knowledge is little discussed in the literature – is the difficulty to use these methods when the dimension of the input space is large (say larger than 10, which is frequent in computer experiments or machine learning).

Let us also mention that the computation of $k(X, X)^{-1}y$, for an arbitrary vector $y \in \mathbb{R}^n$ can be performed using iterative algorithms, like the preconditionned conjugate gradient algorithm [Golub and Van Loan, 2012]. Unfortunately, the algorithms need to be run many times when a posterior variance – involving the computation of $k(X, X)^{-1}k(X, x_i)$ – needs to be computed for a large set of prediction points.

The method proposed in this paper belongs to the so-called "mixture of experts" family. The latter relies on the aggregation of sub-models based on subsets of the data which make them easy to compute. This kind of methods offers a great flexibility since it can be applied with any covariance function and in large dimension while retaining the interpolation property. Some existing "mixture of experts" methods are product of experts [Hinton, 2002], and the (robust) Bayesian committee machine [Tresp, 2000, Deisenroth and Ng, 2015]. All these methods are based on a similar approach: for a given point x, each sub-model provides its own prediction (a mean and a variance) and these predictions are then merged into one single mean and prediction variance. The differences between these methods lie in how to aggregate the predictions made by each sub-model. It shall be noted that aggregating expert opinions is the topic of *consensus statistical methods* (sometimes referred to as *opinion synthesis* or *averaging methods*), where probability distributions representing expert opinions are joined together. Early references are [Winkler, 1968, Winkler, 1981]. A detailed review and an annotated bibliography is given in [Genest and Zidek, 1986] (see also [Satopää et al., 2016, Ranjan and Gneiting, 2010] for

recent related developments). From a probabilistic perspective, usual mixture of experts methods assume that there is some (conditional) independence between the sub-models. Although this kind of hypothesis leads to efficient computations, it is often violated in practice and may lead to poor predictions as illustrated in [Samo and Roberts, 2016]. Furthermore, these methods only provide pointwise confidence intervals instead of a full Gaussian process posterior distribution.

Since our method is part of the mixture of experts framework, it benefits from the properties of the mixture of experts techniques: it does not require the data to be on a grid, the predictions can interpolate the observations and it can be applied to data with small or large scale dependencies regardless of the input space dimension. Compared to other mixtures of experts, we relax the usually made independence assumption so that the prediction takes into account all n^2 pairwise cross-covariances between observations. We show that this addresses two main pitfalls of usual mixture of experts. First, the predictions are more accurate. Second, the theoretical consistency is ensured whereas it is not the case for the product of experts and the Bayesian committee machine methods. The detailed proofs of the later are out of the scope of this paper and we refer the interested reader to [Bachoc et al., 2017] for further details. The proposed method remains computationally affordable: predictions are performed in a few seconds for $n = 10^4$ and a few minutes for $n = 10^5$ using a standard laptop and the proposed online implementation. Finally, the prediction method comes with a naturally associated inference procedure, which is based on cross validation errors.

The proposed method is presented in Section 2. In Section 3, we introduce an iterative scheme for nesting the predictors derived previously. A procedure for estimating the parameters of models is then given in Section 4. Finally, Section 5 compares the method with state-of-the-art aggregation methods on both a simulated dataset and an industrial case study.

2 Pointwise aggregation of experts

Let us now address in more details the framework of this article. The method is based on the aggregation of sub-models defined on smaller subsets of points. Let $X_1, \ldots, X_p$ be subvectors of the vector of observations input points X, it is thus possible to define p associated sub-models (or *experts*) $M_1, \ldots, M_p$. For example, the sub-model M_i can be a Gaussian process regression model based on a subset of the data

$$M_i(x) = \mathrm{E}\left[Y(x)|Y(X_i)\right] = k(x, X_i)k(X_i, X_i)^{-1}Y(X_i)\,, \tag{2}$$

however, we make no Gaussian assumption in this section. For a given prediction point $x \in D$, the p sub-models predictions are gathered into a $p \times 1$ vector $M(x) = (M_1(x) \ldots, M_p(x))^t$. The random column vector $(M_1(x), \ldots, M_p(x), Y(x))^t$ is supposed to be centered with finite first two moments and we consider that both the $p \times 1$ covariance vector $k_M(x) = \mathrm{Cov}\left[M(x), Y(x)\right]$ and the $p \times p$ covariance matrix $K_M(x) = \mathrm{Cov}\left[M(x), M(x)\right]$ are given. Sub-models aggregation (or mixture of experts) aims at merging all the pointwise sub-models $M_1(x), \ldots, M_p(x)$ into one unique pointwise predictor $M_{\mathcal{A}}(x)$ of $Y(x)$. We propose the following aggregation:

Definition 1 (Sub-models aggregation)**.** *For a given point $x \in D$, let $M_i(x)$, $i \in \mathcal{A} = \{1, \ldots, p\}$ be sub-models with covariance matrix $K_M(x)$. Then, when $K_M(x)$ is invertible, we define the sub-model aggregation as:*

$$M_{\mathcal{A}}(x) = k_M(x)^t K_M(x)^{-1} M(x). \tag{3}$$

In practice, the invertibility condition on $K_M(x)$ can be avoided by using matrices pseudo-inverses. Given the vector of observations $M(x) = m(x)$, the associated prediction is

$$m_{\mathcal{A}}(x) = k_M(x)^t K_M(x)^{-1} m(x). \tag{4}$$

Notice that we are here aggregating random variables rather than their distributions. For dependent non-elliptical random variables, expressing the probability density function of $M_{\mathcal{A}}(x)$ as a function

of each expert density $M_i(x)$ is not straightforward. This difference in the approaches implies that the proposed method differs from usual consensus aggregations. For example, aggregating random variables allows to specify the correlations between the aggregated prediction and the experts whereas aggregating expert distributions into a univariate prediction distribution does not characterize uniquely these correlations.

Proposition 1 (BLUP)**.** *$M_{\mathcal{A}}(x)$ is the best linear unbiased predictor of $Y(x)$ that writes $\sum_{i\in\mathcal{A}} \alpha_i(x) M_i(x)$. The mean squared error $v_{\mathcal{A}}(x) = \mathrm{E}\left[(Y(x) - M_{\mathcal{A}}(x))^2\right]$ writes*

$$v_{\mathcal{A}}(x) = k(x,x) - k_M(x)^t K_M(x)^{-1} k_M(x)\,. \tag{5}$$

The coefficients $\{\alpha_i(x), i \in \mathcal{A}\}$ are given by the vector $\alpha = k_M(x)^t K_M(x)^{-1}$.

Proof. The standard proof applies: The square error writes $\mathrm{E}\left[(Y(x) - \alpha^t M(x))^2\right] = k(x,x) - 2\alpha^t k_M(x) + \alpha^t K_M(x)\alpha$. The value of α^* minimising it can be found by differentiation: $-2k_M(x) + 2\alpha^* K_M(x) = 0$ which leads to $\alpha^* = K_M(x)^{-1}k_M(x)$. Then, $v_{\mathcal{A}}(x) = k(x,x) - 2\alpha^{*t}k_M(x) + \alpha^{*t}K_M(x)\alpha^*$ and the result holds. □

Proposition 2 (Basic properties)**.** *Let x be a given prediction point in D.*

(i) Linear case: if $M(x)$ is linear in $Y(X)$, i.e. if there exists a $p \times n$ deterministic matrix $\Lambda(x)$ such that $M(x) = \Lambda(x)Y(X)$ and if $\Lambda(x)k(X,X)\Lambda(x)^t$ is invertible, then $M_{\mathcal{A}}(x)$ is linear in $Y(X)$ with

$$\begin{cases} M_{\mathcal{A}}(x) &= \lambda_{\mathcal{A}}(x)^t Y(X)\,, \\ v_{\mathcal{A}}(x) &= k(x,x) - \lambda_{\mathcal{A}}(x)^t k(X,x)\,. \end{cases} \tag{6}$$

where $\lambda_{\mathcal{A}}(x)^t = k(x,X)\Lambda(x)^t\left(\Lambda(x)k(X,X)\Lambda(x)^t\right)^{-1}\Lambda(x)$.

(ii) Interpolation case: if M interpolates Y at X, i.e. if for any component x_k of the vector X there is at least one index $i_k \in \mathcal{A}$ such that $M_{i_k}(x_k) = Y(x_k)$, and if $K_M(x_k)$ is invertible for any component x_k of X, then $M_{\mathcal{A}}$ is also interpolating, i.e.

$$\begin{cases} M_{\mathcal{A}}(X) &= Y(X)\,, \\ v_{\mathcal{A}}(X) &= 0_n\,, \end{cases} \tag{7}$$

where 0_n is a $n \times 1$ vector with entries 0. This property can be extended when some $K_M(x_k)$ are not invertible by using pseudo-inverse in place of matrix inverse in Definition 1.

(iii) Gaussian case: if the joint distribution $(M(x), Y(x))$ is multivariate normal, then the conditional distribution of $Y(x)$ given $M(x)$ is normal with moments

$$\begin{cases} \mathrm{E}\left[Y(x) | M_i(x), i \in \mathcal{A}\right] = M_{\mathcal{A}}(x)\,, \\ \mathrm{V}\left[Y(x) | M_i(x), i \in \mathcal{A}\right] = v_{\mathcal{A}}(x)\,. \end{cases} \tag{8}$$

Proof. Linearity directly derives from $k_M(x) = \Lambda(x)k(X,x)$ and $K_M(x) = \Lambda(x)K(X,X)\Lambda(x)^t$.
Interpolation: Let $k \in \{1,\ldots,n\}$, and $i \in \mathcal{A}$ be an index such that $M_i(x_k) = Y(x_k)$. As $K_M(x_k) = \mathrm{Cov}\left[M(x_k), M(x_k)\right]$, the i^{th} line of $K_M(x_k)$ is equal to $\mathrm{Cov}\left[M_i(x_k), M(x_k)\right] = \mathrm{Cov}\left[Y(x_k), M(x_k)\right] = k_M(x_k)^t$. Setting e_i the p dimensional vector having entries 0 except on its i^{th} component, it is thus clear that ${e_i}^t K_M(x_k) = k_M(x_k)^t$. As $K_M(x_k)$ is assumed to be invertible, then ${e_i}^t = k_M(x_k)^t K_M(x_k)^{-1}$, so that $M_{\mathcal{A}}(x_k) = k_M(x_k)^t K_M(x_k)^{-1} M(x_k) = {e_i}^t M(x_k) = M_i(x_k) = Y(x_k)$. This result can be plugged into the definition of $v_{\mathcal{A}}$ to obtain the second part of Eq. (7): $v_{\mathcal{A}}(x_k) = \mathrm{E}\left[(Y(x_k) - M_{\mathcal{A}}(x_k))^2\right] = 0$.
Finally the Gaussian case can be proved directly by applying the usual multivariate normal conditioning formula. □

Let us assume here that conditions in items (i) and (ii) of Proposition 2 are satisfied, that is that $M(x)$ is linear in $Y(X)$, and that $M_{\mathcal{A}}(X) = Y(X)$. Then, the proposed aggregation method also benefits from several other interesting properties:

- First, the aggregation can be seen as an exact conditional process for a slightly different prior distribution on Y. One can indeed define a process $Y_{\mathcal{A}}$ as $Y_{\mathcal{A}} = M_{\mathcal{A}} + \varepsilon'_{\mathcal{A}}$ where $\varepsilon'_{\mathcal{A}}$ is an independent replicate of $Y - M_{\mathcal{A}}$ and with $M_{\mathcal{A}}$ as in (3). One can then show that $Y_{\mathcal{A}}(X) = Y(X)$ and
$$\begin{cases} M_{\mathcal{A}}(x) = \mathrm{E}\left[Y_{\mathcal{A}}(x)|Y_{\mathcal{A}}(X)\right] , \\ v_{\mathcal{A}}(x) = \mathrm{V}\left[Y_{\mathcal{A}}(x)|Y_{\mathcal{A}}(X)\right] . \end{cases}$$
Denoting $k_{\mathcal{A}}$ the covariance function of $Y_{\mathcal{A}}$, one can also show that $k_{\mathcal{A}}(x,x) = k(x,x)$ for all $x \in D$ and $k_{\mathcal{A}}(X,X) = k(X,X)$.

- Second, the error between the aggregated model and the full model of Equation (1) can be bounded. For any norm $\|.\|$, one can show that there exists some constants $\lambda, \mu \in \mathbb{R}^+$ such that
$$\begin{cases} |M_{\mathcal{A}}(x) - M_{full}(x)| & \leq & \lambda \|k(X,x)\| \|Y(X)\| , \\ |v_{\mathcal{A}}(x) - v_{full}(x)| & \leq & \mu \|k(X,x)\|^2 . \end{cases}$$
One can also show that the differences between the full and aggregated models write as norm differences, where $\|u\|_K^2 = u^t k(X,X)^{-1} u$:
$$\begin{cases} \mathrm{E}\left[(M_{\mathcal{A}}(x) - M_{full}(x))^2\right] = \|k(X,x) - k_{\mathcal{A}}(X,x)\|_K^2 , \\ v_{\mathcal{A}}(x) - v_{full}(x) = \|k(X,x)\|_K^2 - \|k_{\mathcal{A}}(X,x)\|_K^2 . \end{cases}$$

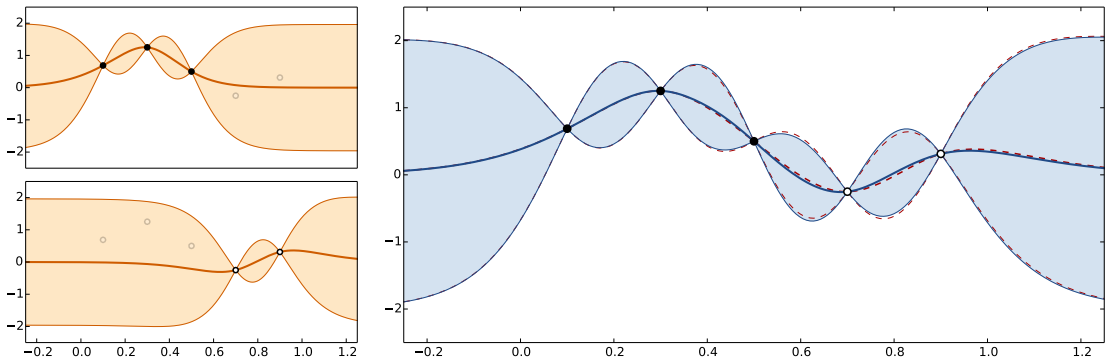
- Third, contrarily to several other aggregation methods, when sub-models are informative enough, the difference between the aggregated model and the full model vanishes: when $M(x) = \Lambda(x) Y(X)$ where $\Lambda(x)$ is a $n \times n$ matrix with full rank, then $Y_{\mathcal{A}} \overset{law}{=} Y$ and $Y_{\mathcal{A}}|Y_{\mathcal{A}}(X) \overset{law}{=} Y|Y(X)$. Furthermore, in this full-information case,
$$\begin{cases} M_{\mathcal{A}}(x) & = & M_{full}(x) , \\ v_{\mathcal{A}}(x) & = & v_{full}(x) . \end{cases}$$

- Finally, in the Gaussian case and under some supplementary conditions, it can be proven that, contrarily to several other aggregation methods, the proposed method is consistent when the number of observation points tends toward infinity. Let $(x_{ni})_{1 \leq i \leq n, n \in \mathbb{N}}$ be a triangular array of observation points such that for all $x \in D$, $\lim_{n \to \infty} \min_{i=1,\ldots,n} ||x_{ni} - x|| = 0$. For $n \in \mathbb{N}$, let $X = (x_{n1}, ..., x_{nn})^t$, let $M_1(x), ..., M_{p_n}(x)$ be any collection of p_n simple Kriging predictors based on respective design points $X_1, \ldots, X_{p_n}$ where $\cup_{i=1}^{p_n} X_i = X$ (with a slight abuse of notation), then
$$\sup_{x \in D} \mathbb{E}\left(\left(Y(x) - M_{\mathcal{A}}(x)\right)^2 \right) \to_{n \to \infty} 0.$$
One can also exhibit non-consistency results for other aggregation methods of the literature that do not use covariances between sub-models.

The full development of these properties is out of the scope of this paper and we dedicate a separate article to detail them [Bachoc et al., 2017].

We now illustrate our aggregation method with two simple examples.

Example 1 (Gaussian process regression aggregation)**.** *In this example, we set $D = \mathbb{R}$ and we approximate the function $f(x) = \sin(2\pi x) + x$ based on a set of five observation points in D: $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. These observations are gathered in two column vectors $X_1 = (0.1, 0.3, 0.5)$ and $X_2 = (0.7, 0.9)$. We use*



(a) sub-models to aggregate (b) aggregated model (solid lines) and full model (dashed lines)

Figure 1: Example of aggregation of two Gaussian process regression models. For each model, we represent the predicted mean and 95% confidence intervals.

as prior a centered Gaussian process Y with squared exponential covariance $k(x,x') = \exp\left(-12.5(x-x')^2\right)$ in order to build two Kriging sub-models, for $i \in \{1,2\}$:

$$\begin{cases} M_i(x) = \mathrm{E}\left[Y(x)|Y(X_i)\right] = k(x,X_i)k(X_i,X_i)^{-1}Y(X_i)\,, \\ m_i(x) = \mathrm{E}\left[Y(x)|Y(X_i) = f(X_i)\right] = k(x,X_i)k(X_i,X_i)^{-1}f(X_i)\,. \end{cases} \tag{9}$$

The expressions required to compute $M_{\mathcal{A}}$ as defined in Eq. (3) are for $i,j \in \{1,2\}$:

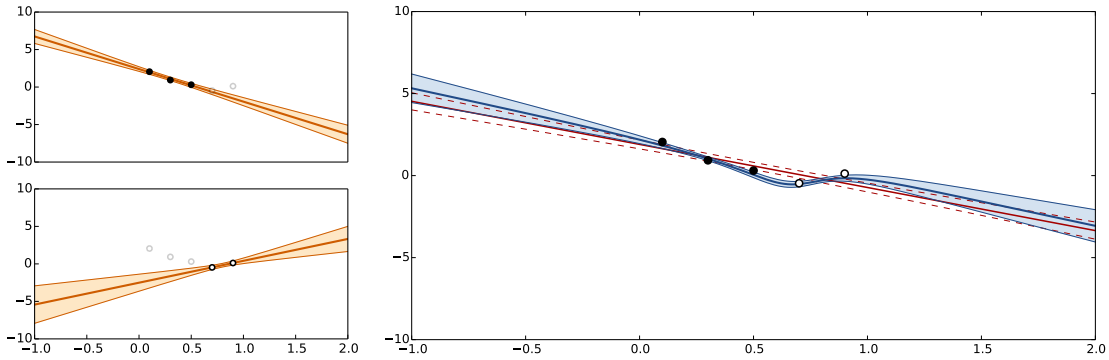
$$\begin{cases} \left(k_M(x)\right)_i = \mathrm{Cov}\left[M_i(x), Y(x)\right] = k(x,X_i)k(X_i,X_i)^{-1}k(X_i,x)\,, \\ \left(K_M(x)\right)_{i,j} = \mathrm{Cov}\left[M_i(x), M_j(x)\right] = k(x,X_i)k(X_i,X_i)^{-1}k(X_i,X_j)k(X_j,X_j)^{-1}k(X_j,x)\,. \end{cases} \tag{10}$$

Recall $m_{full}(x) = \mathrm{E}\left[Y(x)|Y(X) = f(X)\right]$ and $v_{full}(x) = \mathrm{V}\left[Y(x)|Y(X) = f(X)\right]$, as it can be seen in Figure 1, the resulting model $m_{\mathcal{A}}$ appears to be a very good approximation of m_{full} and there is only a slight difference between prediction variances $v_{\mathcal{A}}$ and v_{full} on this example.

Example 2 (Linear regression aggregation)**.** *In this distribution-free example, we set $D = \mathbb{R}$ and we consider the process $Y(x) = \varepsilon_1 + \varepsilon_2 x$ where ε_1 and ε_2 are independent centered random variables with unit variance. Y is thus centered with covariance $k(x,x') = 1 + xx'$. Furthermore, we consider that Y is corrupted by some observation noise $Y_{obs}(x) = Y(x) + \varepsilon_3(x)$ where $\varepsilon_3(x)$ is an independent white noise process with covariance $k_3(x,x') = \mathbf{1}_{\{x=x'\}}$. Note that we only make assumptions on the first two moments of ε_1, ε_2 or $\varepsilon_3(x)$ but not on their laws. We introduce five observation points gathered in two column vectors: $X_1 = (0.1, 0.3, 0.5)^t$ and $X_2 = (0.7, 0.9)^t$ and their associated outputs $y_1 = (2.05, 0.93, 0.31)^t$ and $y_2 = (-0.47, 0.12)^t$. The linear regression sub-models, obtained by square error minimization, are $M_i(x) = k(x,X_i)(k(X_i,X_i) + I_d)^{-1}Y_{obs}(X_i)$, $i \in \{1,2\}$, where I_d stands for the identity matrix. Resulting covariances $\mathrm{Cov}\left[M_i(x), Y(x)\right]$, $\mathrm{Cov}\left[M_i(x), M_j(x)\right]$ and aggregated model $M_{\mathcal{A}}(x)$, $v_{\mathcal{A}}(x)$ of Eq. (8) are then easily obtained. The resulting model is illustrated in Figure 2.*

3 Iterative scheme

In the previous sections, we have seen how to aggregate sub-models $M_1, \ldots, M_p$ into one unique aggregated value $M_{\mathcal{A}}$. Now, starting from the same sub-models, one can imagine creating several aggregated values, $M_{\mathcal{A}_1}, \ldots, M_{\mathcal{A}_s}$, each of them based on a subset of $\{M_1, \ldots, M_p\}$. One can show that these aggregated values can themselves be aggregated. This makes possible the construction of an iterative algorithm that merges sub-models at successive steps, according to a tree structure. Such tree-based schemes are sometimes used to reduce the complexity of models, see e.g. [Tzeng et al., 2005], or to allow parallel computing [Wei et al., 2015].



(a) sub-models to aggregate (b) aggregated model (solid lines) and full model (dashed lines)

Figure 2: Example of aggregation of two linear regression sub-models. Exhibited confidence bands correspond to a difference to mean value of two standard deviations.

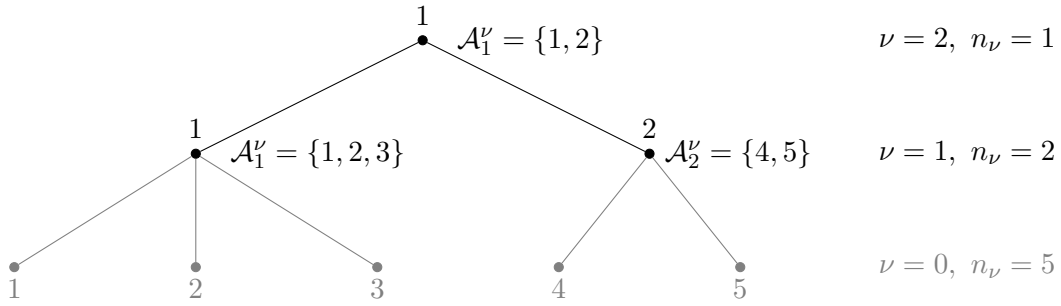


Figure 3: One aggregation tree with height $\bar{\nu} = 2$, $n_0 = 5$ initial leave nodes (observation points) and $n_1 = 2$ sub-models.

The aim of this section is to give a generic algorithm for aggregating sub-models according to a tree structure and to show that the choice of the tree structure helps partially reducing the complexity of the algorithm. It also aims at giving perspectives for further large reduction of the global complexity.

Let us introduce some notations. The total height (i.e number of layers) of the tree is denoted $\bar{\nu}$ and the number of node of a layer $\nu \in \{1, \dots, \bar{\nu}\}$ is n_ν. We associate to each node (say node i in layer ν) a sub-model M_i^ν corresponding to the aggregation of its child node sub-models. In other words, M_i^ν is the aggregation of $\{M_k^{\nu-1},\ k \in \mathcal{A}_i^\nu\}$ where $\mathcal{A}_i^\nu$ is the set of children of node i in layer ν. These notations are summarized in Figure 3 which details the tree associated with Example 1. In practice, there will be one *root node* ($n_{\bar{\nu}} = 1$) and each node will have at least one parent: $\cup_{i=1,\dots,n_\nu} \mathcal{A}_i^\nu = \{1, \dots, n_{\nu-1}\}$. Typically, the sets $\mathcal{A}_i^\nu$, $i = 1, \dots, n_\nu$, are a partition of $\{1, \dots, n_{\nu-1}\}$ but this assumption is not required and a child node may have several parents (which can generate a lattice rather than a tree).

3.1 Two-Layer aggregation

We discuss in this section the tree structure associated with the case $\bar{\nu} = 2$ as per the previous examples. With such settings, the first step consists in calculating the initial sub-models $M_1^1, \dots, M_p^1$ of the layer $\nu = 1$ and the second one is to aggregate all sub-models of layer $\nu = 1$ into one unique predictor M_1^2 (see for example Figure 3). This aggregation is obtained by direct application of Definition 1.

In practice the sub-models can be any covariates, like gradients, non-Gaussian underlying factors or even black-box responses, as soon as cross-covariances and covariances with $Y(x)$ are known. When

sub-models are calculated from direct observations $Y(x_1), \ldots, Y(x_n)$, the number of leave nodes at layer $\nu = 0$ is $n_0 = n$. In further numerical illustrations of Section 5, the sub-models M_i^1 are simple Kriging predictors of $Y(x)$, with for $i = 1, \ldots, p$,

$$\left\{ \begin{array}{lcl} M_i^1(x) & = & k(x, X_i)k(X_i, X_i)^{-1}Y(X_i)\,, \\ \mathrm{Cov}\left[M_i^1(x), Y(x)\right] & = & k(x, X_i)k(X_i, X_i)^{-1}k(X_i, x)\,, \\ \mathrm{Cov}\left[M_i^1(x), M_j^1(x)\right] & = & k(x, X_i)k(X_i, X_i)^{-1}k(X_i, X_j)k(X_j, X_j)^{-1}k(X_j, x)\,. \end{array} \right. \tag{11}$$

With these particular simple Kriging initial sub-models, the layer $\nu = 1$ corresponds to the aggregation of covariates $M_i^0(x) = Y(x_i)$ at the previous layer $\nu = 0$, $i = 1, \ldots, n$.

3.2 Multiple Layer aggregation

In order to extend the two-layer settings, one needs to compute covariances among aggregated sub-models. The following proposition gives covariances between aggregated models of a given layer.

Proposition 3 (aggregated models covariances)**.** *Let us consider a layer $\nu \geq 1$ and given aggregated models $M_1^\nu(x), \ldots, M_{n_\nu}^\nu(x)$. Assume that the following covariances $(k^\nu(x))_i = \mathrm{Cov}\left[M_i^\nu(x), Y(x)\right]$ and $(K^\nu(x))_{ij} = \mathrm{Cov}\left[M_i^\nu(x), M_j^\nu(x)\right]$ are given, $i, j \in \{1, \ldots, n_\nu\}$. Let $n_{\nu+1} \geq 1$ be a number of new aggregated values. Consider subsets $\mathcal{A}_i^{\nu+1}$ of $\{1, \ldots, n_\nu\}$, $i = 1, \ldots, n_{\nu+1}$, and assume that $M_i^{\nu+1}(x)$ is the aggregation of $M_k^\nu(x)$, $k \in \mathcal{A}_i^{\nu+1}$. Then*

$$\left\{ \begin{array}{lcl} (M^{\nu+1}(x))_i & = & \alpha_i^{\nu+1}(x)^t \left(M^\nu(x)_{[\mathcal{A}_i^{\nu+1}]}\right)\,, \\ \mathrm{Cov}\left[M_i^{\nu+1}(x), Y(x)\right] & = & \alpha_i^{\nu+1}(x)^t \left(k^\nu(x)_{[\mathcal{A}_i^{\nu+1}]}\right)\,, \\ \mathrm{Cov}\left[M_i^{\nu+1}(x), M_j^{\nu+1}(x)\right] & = & \alpha_i^{\nu+1}(x)^t \left(K^\nu(x)_{[\mathcal{A}_i^{\nu+1}, \mathcal{A}_j^{\nu+1}]}\right) \alpha_j^{\nu+1}(x)\,, \end{array} \right. \tag{12}$$

where the vectors of optimal weights are $\alpha_i^{\nu+1}(x) = \left(K^\nu_{[\mathcal{A}_i^{\nu+1}, \mathcal{A}_i^{\nu+1}]}\right)^{-1} \left(k^\nu(x)_{[\mathcal{A}_i^{\nu+1}]}\right)$ and where $k^\nu(x)_{[\mathcal{A}_i^{\nu+1}]}$ corresponds to the sub-vector of $k^\nu(x)$ of indices in $\mathcal{A}_i^{\nu+1}$ and similarly for $M^\nu(x)_{[\mathcal{A}_i^{\nu+1}]}$ and the sub-matrix $K^\nu(x)_{[\mathcal{A}_i^{\nu+1}, \mathcal{A}_i^{\nu+1}]}$, which is assumed to be invertible.
Furthermore, $\mathrm{Cov}\left[M_i^{\nu+1}(x), Y(x)\right] = \mathrm{Cov}\left[M_i^{\nu+1}(x), M_i^{\nu+1}(x)\right]$.

Proof. This follows immediately from Definition 1: as the aggregated values are linear expressions, the calculation of their covariances is straightforward. The last equality is simply obtained by inserting the value of $\alpha_i^{\nu+1}(x)$ into the expression of $\mathrm{Cov}\left[M_i^{\nu+1}(x), M_i^{\nu+1}(x)\right]$. □

The following algorithm, which is a generic algorithm for aggregating sub-models according to a tree structure, is based on an iterative use of the previous proposition. It is given for one prediction point $x \in D$ and it assumes that the sub-models are already calculated, starting directly from layer 1. This allows a large variety of sub-models, and avoids the storage of the possibly large covariance matrix $K^0(x)$. Its outputs are the final scalar aggregated model, $M_{\bar{\nu}}(x)$, and the scalar covariance $K_{\bar{\nu}}(x)$ from which one deduces the prediction error $\mathrm{E}\left[(Y(x) - M_{\bar{\nu}}(x))^2\right] = k(x, x) - K_{\bar{\nu}}(x)$.

In order to give dimensions in the algorithm and to ease the calculation of complexities, we define c_i^ν as the number of children of the sub-model M_i^ν, $c_i^\nu = \mathrm{card}\,\mathcal{A}_i^\nu$. We also denote $c_{\max} = \max_{\nu,i} c_i^\nu$ the maximal number of children.

Algorithm 1: Nested Kriging algorithm

inputs : M_1, vector of length n_1 (sub-models evaluated at x)
k_1, vector of length n_1 (covariance between $Y(x)$ and sub-models at x)
K_1, matrix of size $n_1 \times n_1$ (covariance between sub-models at x)
$\mathcal{A}$, a list describing the tree structure
outputs: $M_{\bar{\nu}}$, $K_{\bar{\nu}}$

Create vectors M, k of size $c_{\max}$ and matrix K of size $c_{\max} \times c_{\max}$
for $\nu = 2, \ldots, \bar{\nu}$ **do**
  Create vectors M_ν of size n_ν and matrix K_ν of size $n_\nu \times n_\nu$
  for $i = 1, \ldots, n_\nu$ **do**
    Create vector α_i of size c_i^ν
    $M \leftarrow$ subvector of $M_{\nu-1}$ on $\mathcal{A}_i^\nu$
    $K \leftarrow$ submatrix of $K_{\nu-1}$ on $\mathcal{A}_i^\nu$
    if $\nu = 2$ **then** $k \leftarrow k_1$ **else** $k \leftarrow \mathrm{Diag}(K)$
    $\alpha_i \leftarrow K^{-1}k$
    $M_\nu[i] \leftarrow (\alpha_i)^t M$
    $K_\nu[i,i] \leftarrow (\alpha_i)^t k$
    for $j = 1, \ldots, i-1$ **do**
      $K \leftarrow$ submatrix of $K_{\nu-1}$ on $\mathcal{A}_i^\nu \times \mathcal{A}_j^\nu$
      $K_\nu[i,j] \leftarrow (\alpha_i)^t K \alpha_j$
      $K_\nu[j,i] \leftarrow K_\nu[i,j]$
  $M_{\nu-1}$, $K_{\nu-1}$ and all α_i can be deleted

Notice that Algorithm 1 uses the result $(K^{\nu+1}(x))_{ii} = (k^{\nu+1}(x))_i$ from Prop. 3: when we consider aggregated models ($\nu \geq 2$), we do not need to store and compute the vector $k^\nu(x)$ any more. When $\nu = 1$, depending on the initial covariates, $\mathrm{Cov}\left[M_i^1(x), Y(x)\right]$ is not necessarily equal to $\mathrm{Cov}\left[M_i^1(x), M_i^1(x)\right]$ (this is however the case when $M_i^1(x)$ are simple Kriging predictors).

For the sake of clarity, some improvements have been omitted in the algorithm above. For instance, covariances can be stored in triangular matrices, one can store two couples (M_ν, K_ν) instead of $\bar{\nu}$ couples by using objects $M_{(\nu \mod 2)}$ and $K_{(\nu \mod 2)}$. Furthermore, it is quite natural to adapt this algorithm to parallel computing, but this is out of the scope of this article.

3.3 Complexity

We study here the complexity of Algorithm 1 in space (storage footprint) and in time (execution time). For the sake of clarity we consider in this paragraph a simplified tree where n_ν is decreasing in ν and each child has only one parent. This corresponds to the most common structure of trees, without overlapping. Furthermore, at any given level ν, we consider that each node has the same number of children: $c_i^\nu = c_\nu$ for all $i = 1, \ldots, n_\nu$. Such a tree will be called *regular*. In this setting, one easily sees that $n_\nu = \frac{n_{\nu-1}}{c_\nu} = \frac{n}{c_1 \ldots c_\nu}$, $\nu \in \{1, \ldots, \bar{\nu}\}$. Complexities obviously depend on the choice of sub-models, we give here complexities for Kriging sub-models as in Eq. (11), but this can be adapted to other kinds of sub-models.

For one prediction point $x \in D$, we denote by $\mathcal{S}$ the storage footprint of Algorithm 1, and by $\mathcal{C}$ its complexity in time, including sub-models calculation. One can show that in a particular two-layers setting with $\sqrt{n}$ sub-models ($\bar{\nu} = 2$ and $c_1 = c_2 = \sqrt{n}$), a reachable global complexity for q prediction points is (see assumptions below and expression details in the proof of Proposition 4)

$$\mathcal{S} = O(n) \quad \text{and} \quad q\mathcal{C} = O(n^2 q)\,. \tag{13}$$

This is to be compared with $O(n^3)+O(n^2q)$ for the same prediction with the full model. The aggregation of sub-models can be useful when the number of prediction points is smaller than the number of observations. Notice that the storage needed for q prediction points is the same as for one prediction point, but in some cases (as for leave-one-out errors calculation), it is worth using a $O(nq)$ storage to avoid recalculations of some quantities.

We now detail chosen assumptions on the calculation of $\mathcal{S}$ and $\mathcal{C}$, and study the impact of the tree structure on these quantities. For one prediction point $x \in D$, including sub-models calculation, the complexity in time can be decomposed into $\mathcal{C} = \mathcal{C}_{\text{cov}} + \mathcal{C}_\alpha + \mathcal{C}_\beta$, where

- $\mathcal{C}_{\text{cov}}$ is the complexity for computing all cross covariances among initial design points, which does not depend on the tree structure (neither on the number of prediction points).
- $\mathcal{C}_\alpha$ is the complexity for building all aggregation predictors, i.e. the sum over ν, i of all operations in the i-loop in Algorithm 1 (excluding operations in the j-loop).
- $\mathcal{C}_\beta$ is the complexity for building the covariance matrices among these predictors, i.e. the sum over ν, i, j of all operations in the j-loop in Algorithm 1.

We assume here that there exists two constants $\alpha > 0$ and $\beta > 0$ such that the complexity of operations inside the i-loop (excluding those of the j-loop) is αc_ν^3, and the complexity of operations inside the j-loop is βc_ν^2. Despite perfectible, this assumption follows from the fact that one usually considers that the complexity of $c_\nu \times c_\nu$ matrix inversion is $O(c_\nu^3)$ and the complexity of matrix-vector multiplication is $O(c_\nu^2)$. We also assume that the tree height $\bar{\nu}$ is finite, and that all numbers of children c_ν tend to $+\infty$ as n tends to $+\infty$. This excludes for example binary trees, but makes assumptions on complexities more reliable. Under these assumptions, the following proposition details how the tree structure affects the complexities.

Proposition 4 (Complexities)**.** *The following storage footprint $\mathcal{S}$ and complexities $\mathcal{C}_\alpha$, $\mathcal{C}_\beta$ hold for the respective tree structures, when the number of observations n tends to ∞.*

(i) The two-layer equilibrated $\sqrt{n}$-tree, where $p = c_1 = c_2 = \sqrt{n}$, $\bar{\nu} = 2$, is the optimal storage footprint tree, and

$$\mathcal{S} = O(n)\,, \qquad \mathcal{C}_\alpha \sim \alpha n^2\,, \qquad \mathcal{C}_\beta \sim \frac{\beta}{2} n^2\,. \tag{14}$$

(ii) The $\bar{\nu}$-layer equilibrated $\sqrt[\bar{\nu}]{n}$-tree, where $c_1 = \cdots = c_{\bar{\nu}} = \sqrt[\bar{\nu}]{n}$, $\bar{\nu} \geq 2$, is such that

$$\mathcal{S} = O(n^{2-2/\bar{\nu}})\,, \qquad \mathcal{C}_\alpha \sim \alpha n^{1+\frac{2}{\bar{\nu}}}\,, \qquad \mathcal{C}_\beta \sim \frac{\beta}{2} n^2\,. \tag{15}$$

(iii) The optimal complexity tree is defined as the regular tree structure that minimizes $\mathcal{C}_\alpha$, as it is not possible to reduce $\mathcal{C}_\beta$ to orders lower than $O(n^2)$. This tree is such that

$$\mathcal{S} = O\left(n^{2-\frac{1}{\delta^{\bar{\nu}}-1}}\right)\,, \qquad \mathcal{C}_\alpha \sim \gamma\alpha n^{1+\frac{1}{\delta^{\bar{\nu}}-1}}\,, \qquad \mathcal{C}_\beta \sim \frac{\beta}{2} n^2\,, \tag{16}$$

with $\delta = \frac{3}{2}$ and $\gamma = \frac{27}{4}\delta^{-\frac{\bar{\nu}}{\delta^{\bar{\nu}}-1}}\left(1-\delta^{-\bar{\nu}}\right)$. This tree is obtained for $c_\nu = \delta\left(\delta^{-\bar{\nu}} n\right)^{\frac{\delta^{(\nu-1)}}{2(\delta^{\bar{\nu}}-1)}}$, $\nu = 1, \ldots, \bar{\nu}$. In a particular two-layers setting one gets $c_1 = \left(\frac{3}{2}\right)^{1/5} n^{2/5}$ and $c_2 = \left(\frac{3}{2}\right)^{-1/5} n^{3/5}$, which leads to $\mathcal{C}_\alpha = \gamma\alpha n^{9/5}$ and $\mathcal{C}_\beta = \frac{\beta}{2}n^2 - \frac{\beta}{2}\left(\frac{3}{2}\right)^{\frac{1}{5}} n^{\frac{7}{5}}$, where $\gamma = (\frac{2}{3})^{-2/5} + (\frac{2}{3})^{3/5} \simeq 1.96$.

Proof. The details of the proof are given in Appendix A. □

We have seen that for q prediction points and n observations, a reachable complexity of the algorithm is $O(n^2q)$, which is less than $O(n^3) + O(n^2q)$ for the same prediction with the full model, when $q < n$.

More precisely, we have shown that the choice of the tree structure helps partially reducing the complexity of the algorithm. Indeed, a large tree height $\bar{\nu}$ largely reduces the complexity $\mathcal{C}_\alpha$ of matrix inversions in the algorithm. However, $\mathcal{C}_\beta$ cannot be reduced and one can expect a maximal complexity reduction factor of $\frac{\beta}{2\alpha+\beta}$ when using an optimal tree, compared to the equilibrated two-layers $\sqrt{n}$-tree. One shall however keep in mind that a lower complexity can lead to larger prediction errors or larger storage footprint.

As a perspective, approximating cross-covariances between aggregated models would allow to reduce $\mathcal{C}_\beta$ to the same order as $\mathcal{C}_\alpha$, which approaches $O(n)$ when $\bar{\nu}$ is large. This thus gives perspectives for further large reduction of the global complexity, which are let to future work.

At last, several parts of the algorithm can be computed in parallel execution threads. This is an interesting feature since sub-models computation at any layer can also be distributed.

4 Parameter estimation

Consider a set of covariance functions $\{\sigma^2 k_\theta, \sigma^2 \geq 0, \theta \in \Theta\}$ where k_θ is a correlation function from $D \times D$ into $[-1, 1]$ depending on some parameters θ such as length-scales. In this section, we address the problem of selecting the value of σ^2 and θ from the input observation points in X and the observation vector $f(X)$. The mean predictor $m_{\mathcal{A}}$ depends only on θ so it will be written $m_{\mathcal{A},\theta}$. The prediction variance is a function of both θ and σ^2. Since it is linear in the latter, the prediction variance is written $\sigma^2 v_{\mathcal{A},\theta}$.

For $1 \leq i \leq n$, let the leave-one-out mean $m_{\mathcal{A},\theta,-i}(x_i)$ be computed as $m_{\mathcal{A},\theta}(x_i)$, but with $X, f(X)$ replaced by $X_{-i}, f(X_{-i})$, where X_{-i} is obtained by removing the ith line of X. Note that the input division $X_1, ..., X_p$ is left unchanged, apart from removing x_i when it appears in $X_1, ..., X_p$. Similarly, the tree structure $\{\mathcal{A}_i^\nu\}$ is left unchanged. We define $\sigma^2 v_{\mathcal{A},\theta,-i}(x_i)$ similarly to $\sigma^2 v_{\mathcal{A},\theta}(x_i)$.

We estimate σ^2 and θ with a two-step leave-one-out procedure similar to that of [Bachoc, 2013]. We first select θ as minimizing the leave-one-out mean square error:

$$\widehat{\theta} \in \operatorname*{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i) - m_{\mathcal{A},\theta,-i}(x_i) \right)^2 . \tag{17}$$

Second, we set σ^2 so that the leave-one-out errors have variance one:

$$\widehat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \frac{\left(f(x_i) - m_{\mathcal{A},\widehat{\theta},-i}(x_i) \right)^2}{v_{\mathcal{A},\widehat{\theta},-i}(x_i).} \tag{18}$$

We implemented an algorithm that computes, for a given covariance parameter θ, the quantities $m_{\mathcal{A},\theta,-i}(x_i)$ and $v_{\mathcal{A},\theta,-i}(x_i)$ for q different points x_i. If the proper storage and precomputations are made, the computational cost is of $O(qn^2)$, which is similar to the cost for predicting at q new locations using the model aggregation procedure presented in this paper. However using precomputations, the algorithm also has a storage cost of $O(nq)$ which excludes using $q = n$ in the case where n is large and prevents computing the right-hand side of Equation (17) exactly. Finally, one may notice that when q points are chosen uniformly, without replacement, in the set of all n points, averaging q leave-one-out mean square error yields an unbiased estimate of the leave-one-out mean square error, and can be seen as an approximation of the latter. We thus propose to solve the optimization problem (17) with a stochastic gradient descent algorithm described in Chapter 5 of [Bhatnagar et al., 2013]. At each step of the gradient descent, the projection of the gradient of (17) on a random direction is approximated by a finite difference. The algorithm is as follows.

Algorithm 2: Stochastic gradient descent

inputs : θ_0, initial value of θ
$(a_i)_{i\in\mathbb{N}}$, sequence of increment terms for the gradient descent
$(\delta_i)_{i\in\mathbb{N}}$, sequence of step sizes for the finite differences
q, number of leave-one-out predictions
n_{iter}, maximal number of iterations
outputs: $\hat{\theta}$

for $i = 1, ..., n_{iter}$ **do**
Sample a subset $\mathcal{I}_i$ of $\{1, ..., n\}$, uniformly over all the subsets of $\{1, ..., n\}$ with cardinality q.
Sample a m-dimensional vector h_i from a m-dimensional random vector with independent components, each of them taking the values 1 and -1 with probabilities $1/2$.
Let

$$\Delta_i = \frac{1}{2\delta_i}\left(\frac{1}{q}\sum_{j\in\mathcal{I}_i}\left(f(x_j) - m_{\mathcal{A},-j,\theta_{i-1}+\delta_i h_i}(x_j)\right)^2 - \frac{1}{q}\sum_{j\in\mathcal{I}_i}\left(f(x_j) - m_{\mathcal{A},-j,\theta_{i-1}-\delta_i h_i}(x_j)\right)^2\right).$$

Let $\theta_i = \theta_{i-1} - a_i\Delta_i h_i$.
Let $\hat{\theta} = \theta_{n_{iter}}$.

An implementation in R and C++ of both algorithms 1 and 2 is publicly available on the website `http://www.clementchevalier.com/index.php/r-packages`. In practice, the computation cost of q leave-one-out predictions is the sum of a fixed cost – involving in particular the computation of the n^2 covariances $k_\theta(x_i, x_j)$ – and a marginal cost which is proportional to q. When $n = 10,000$, these two summands take comparable values for $q = 100$, which is the setting we use in practice. Following the recommendations in [Bhatnagar et al., 2013], we set $\delta_i = c/(i+1)^\gamma$, with $\gamma = 0.101$. We set $a_i = a/(A+i+1)^\alpha$, with $\alpha = 0.602$ (as suggested in [Bhatnagar et al., 2013]), or $\alpha = 0.2$, or a combination of these two values. Typically we run a first gradient descent with $\alpha = 0.2$, which termination point serves as starting point for a second gradient descent with $\alpha = 0.602$. Good values of a, c and A depend of the application case. In practice, satisfactory results are obtained for $n = 10,000$, $d = 10$ and $p = 100$, with $n_{iter} = 500$, in which case the computation time would be around a few hours on a personal computer with a mono-threaded implementation.

5 Numerical applications

5.1 Comparison with other aggregation methods

We now compare the predictions obtained with various methods when aggregating 15 Kriging sub-models based on two observations each. The test functions are samples of a centered Gaussian process over $[0, 1]$. The compared models are the nested Kriging model introduced in this article, the full model and other methods developed in the literature:

Product of expert (PoE) [Hinton, 2002] is based on the assumption that for a given x, the predictions of each sub-model correspond to independent random variables. As a consequence, the aggregated predicted density for $Y(x)$ is equal to the product of the sub-models densities : $f_{poe}(y) \propto \prod_{i=1}^{p} f_i(y)$ where f_i is the predicted density of $Y(x)$ according to the ith sub-model. The PoE corresponds to the normal model developed in [Winkler, 1981], in the case of independent experts, when the considered covariance matrix is diagonal (see e.g. section 3.2 in the previously cited article and [van Stein et al., 2015]). Some extensions of this method to consensus Monte-Carlo sampling can be found in [Scott et al., 2016].

Generalized product of expert (GPoE). As discussed in [Deisenroth and Ng, 2015], a major drawback of Kriging based PoE is that the prediction variance of the aggregated model decreases when the number of sub-models increases even in regions with no observation points. [Cao and Fleet, 2014]

introduced a variant called generalized product of expert where a weighting term is added to overcome this issue. The prediction is then given by

$$f_{gpoe}(y) \propto \prod_{i=1}^{p} (f_i(y))^{\beta_i}. \tag{19}$$

For this benchmark, the parameters β_i will be set to $1/p$ as recommended in [Deisenroth and Ng, 2015]. Notice that GPoE corresponds exactly to what consensus literature refers to *logarithmic opinion pool*, see e.g. Eq.(3.11) in [Genest and Zidek, 1986].

Bayesian Committee Machine (BCM) has been introduced in [Tresp, 2000] to aggregate Kriging sub-models. It is based on the assumption of conditional independence of the sub-models given the process values at prediction points. The predicted aggregated density is given by

$$f_{bcm}(y) \propto \frac{\prod_{i=1}^{p} f_i(y)}{f_Y(y)^{p-1}}. \tag{20}$$

Robust Bayesian committee machine (RBCM) has been introduced in [Deisenroth and Ng, 2015] to correct some supposed flaws from BCM aggregations in the case where there are only few observations in each sub-model. The predicted aggregated density is given by

$$f_{rbcm}(y) \propto \frac{\prod_{i=1}^{p} (f_i(y))^{\beta_i}}{(f_Y(y))^{-1+\sum_i \beta_i}}, \tag{21}$$

where $\beta_i = \frac{1}{2}[\log(\mathrm{V}[Y(x)]) - \log(v_i(x))]$ with $v_i(x)$ the predicted variance of the i^{th} sub-model at x.

One advantage of these aggregation methods is their very low complexity. However, GPoE, BCM and RBCM can be proven to be inconsistent [Bachoc et al., 2017]. For the sake of comparison, we add two other methods to the benchmark:

Smallest prediction variance (SPV). For a given prediction point x, the aggregation returns the prediction of the sub-model with the lowest prediction variance:

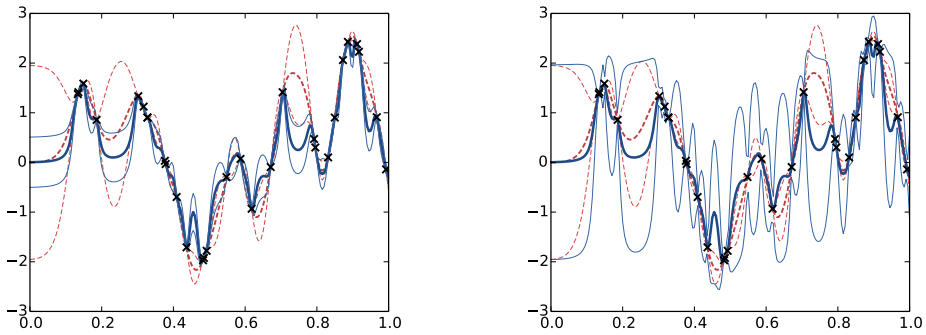
$$f_{spv}(y) = f_k(y) \qquad \text{with } k = \underset{i \in \{1,\dots,p\}}{\operatorname{argmin}}\, v_i(x). \tag{22}$$

Nearest Neighbors (NN). This is not strictly speaking an aggregation method: For a given prediction point x, it consists in a kriging predictor based only on the k observations points that are the closest to x.

These last two methods do not suffer from the inconsistency discussed for the above methods, but they provide discontinuous predictions. This can be an issue for example if the (aggregated) model is used to perform some optimization tasks.

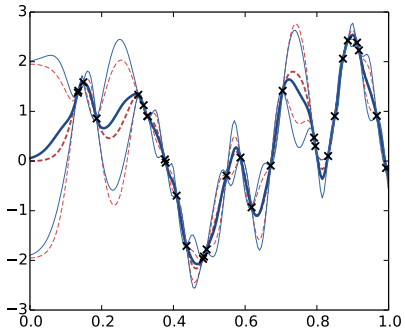
The test functions are given by samples over $[0, 1]$ of a centered Gaussian process Y with a Matérn kernel of smoothness $5/2$. The variance and length-scale parameters of the latter are fixed to $\sigma^2 = 1$ and $\theta = 0.05$. The vector of observation points X consists of 30 random points uniformly distributed on $[0, 1]$ and we consider in this example the aggregation of 15 sub-models based on two points each. Assuming that the observations points are ordered ($x_1 \leq ... \leq x_n$), each sub-model is trained with two consecutive observations points : $\mathcal{A}_1 = \{1, 2\}$, ..., $\mathcal{A}_{15} = \{29, 30\}$. The variance and length-scale parameters of the sub-models are equal to the values used to generate the process samples.

First of all, we will focus on the aggregated models obtained with the different methods for a given sample path and design of experiments X before looking at the distribution of various criteria when replicating the experiment. Figure 4 shows the aggregated models for the aggregation methods described above. On this example, PoE and GPoE appear respectively to be over- and under-confident in their predictions and show a mean prediction that tends too quickly to zero as the prediction point moves away from the observation points. On the other hand, the predictions from other methods seem more reliable and the best approximation is obtained with the proposed nested estimation approach.

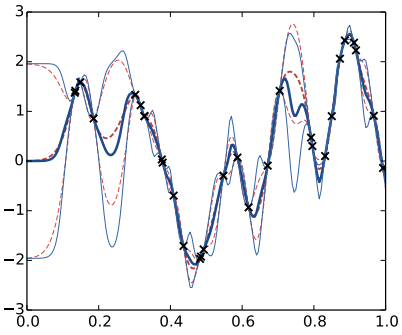


(a) PoE

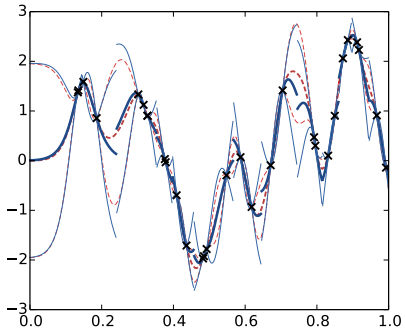
(b) GPoE



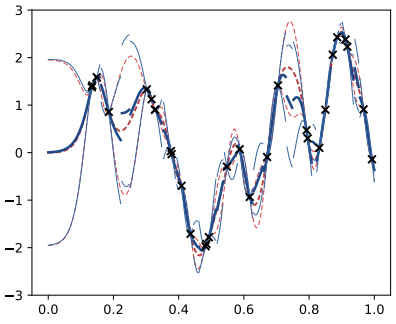
(c) BCM



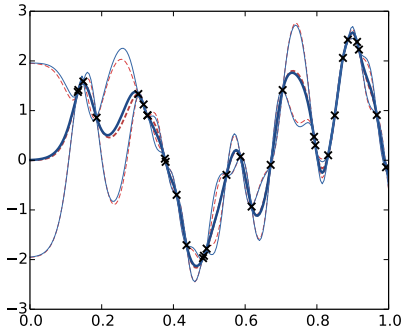
(d) RBCM



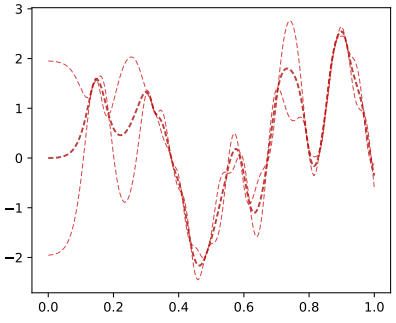
(e) SPV



(f) NN



(g) nested GPR



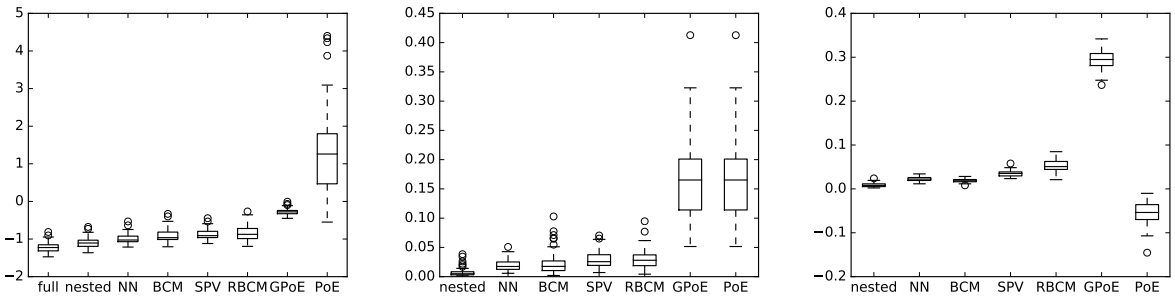
(h) full

Figure 4: Comparison of various aggregation methods. The solid lines correspond to aggregated models (mean and 95% prediction intervals) and the dashed lines indicate the full model predictions (mean and 95% prediction intervals).

This can be confirmed by replicating 50 times the experiment by sampling independently the observation points and the test function. We consider three criteria to quantify the distance between the aggregated model and the full model: the mean square error (MSE) to assess the accuracy of the aggregated mean, the mean variance error MVE for the accuracy of the predicted variance and the mean negative log probability (MNLP) [Williams and Rasmussen, 2006] to quantify the overall distribution fit. Let m, v (resp. m_{full}, v_{full}) denote the mean and variance of the model to be tested (resp. the full model) and let X_t be the vector of test points. These criteria are defined as:

$$\begin{cases} MSE(m, m_{full}, X_t) = \dfrac{1}{n_t}\displaystyle\sum_{i=1}^{n_t}(m(x_{t,i}) - m_{full}(x_{t,i}))^2\,, \\ MVE(v, v_{full}, X_t) = \dfrac{1}{n_t}\displaystyle\sum_{i=1}^{n_t}(v(x_{t,i}) - v_{full}(x_{t,i}))\,, \\ MNLP(m, v, f, X_t) = \dfrac{1}{n_t}\displaystyle\sum_{i=1}^{n_t}\left(\frac{1}{2}\log(2\pi v(x_{t,i})) + \frac{(m(x_{t,i}) - f(x_{t,i}))^2}{2v(x_{t,i})}\right)\,. \end{cases} \tag{23}$$

Figure 5 shows the boxplots of these criteria for 50 replications of the experiments. It appears that the proposed approach gives the best approximation of the full model for the three considered criteria.



(a) MNLP (b) MSE (c) MVE

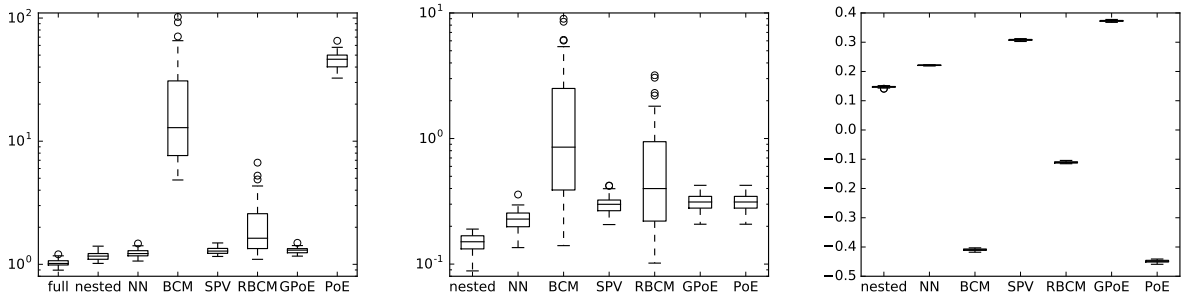
Figure 5: Quality assessment of the aggregated models for 50 test functions. Each test function is a sample from a Gaussian process and in each case 30 observation points are sampled uniformly on $[0, 1]$. The test points vector X_t consists of 101 points regularly spaced from $x_{t,1} = 0$ to $x_{t,101} = 1$.

5.2 Application to a high dimensional input space

We replicate in this section the same experiment as in the previous one but for test functions defined over the unit cube in 100 dimensions. We set the number of training points to $10,000$ and we generate 100 sub-models based on a k-means clustering of the input points. This implies that the average number of points per cluster is 100 so we use the 100 closest observation points in the nearest neighbors method. As previously the test functions are random samples of a centered Gaussian process with squared exponential covariance and we consider two length-scale values to study this parameter influence on the methods to compare: a "short" length-scale $\theta = 2$ for which the full model captures about 50% of the prior variance and a "large" length-scale $\theta = 5$ for which the full model can explain 99% of the prior variance.

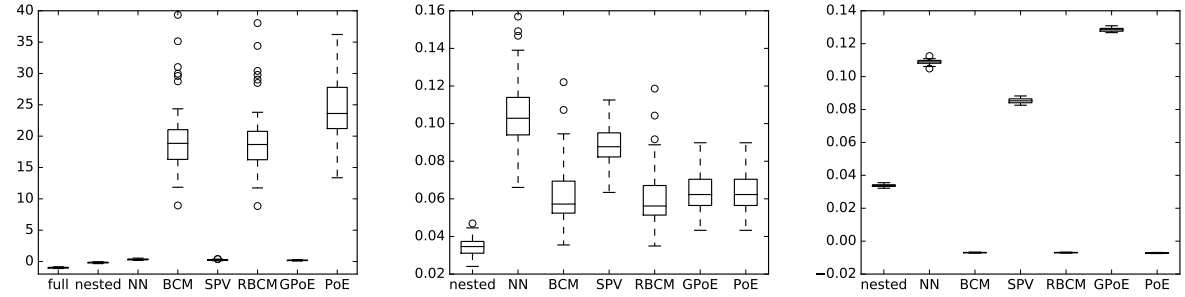
The results of the experiment are displayed in Figures 6 and 7. The first striking observation is that BCM, RBCM and PoE underestimate the variance (since MVEs are negative) and lead to highly overconfident models. Regarding the other approaches, NN, SPV and GPoE seem to provide similar global accuracy although it can be noted that the Nearest Neighbor method mean predictions are inaccurate for large length-scales. This can be explained by the set of influential neighbors being larger than 100 for such values of the length-scales. Finally, it can be seen that the proposed nested

method is the one providing the best approximation of the full model. Compared to the other methods its mean is more accurate and it provides prediction intervals that are smaller than NN, SPV and GPoE while being realistic as shown by the MNLP. This is especially true for large length-scales for which the proposed approach is even more competitive.



(a) MNLP (log scale) (b) MSE (log scale) (c) MVE

Figure 6: Quality assessment of various aggregation methods. The test functions are given by 50 samples from a centered Gaussian process over $[0,1]^{100}$ with squared exponential kernel, unit variance and length-scale $\theta = 2$. The models are built using $10,000$ observations points drawn uniformly in the input space. These input points are gathered into 100 groups using k-means in order to build the sub-models. The test point locations are obtained by sampling uniformly 100 points in the input space.



(a) MNLP (b) MSE (c) MVE

Figure 7: Same settings as in Figure 6, but the length-scale of the test functions is 5.

5.3 Application to a large dataset

In this section, we analyze the performance of the proposed method on a test function with one million observations.

We use two different test functions. The first one is the Hartman6 test function in dimension $d = 6$ (available for example in the DiceKriging R package [Roustant et al., 2012]). The second one in the dimension $d = 18$ is called here Hartman18 test function: it is simply the sum of three Hartman6 test functions, each acting on 6 separated parameters: $\text{Hartman}_{18}(x_{1:18}) = \text{Hartman}_6(x_{1:6}) + \text{Hartman}_6(x_{7:12}) + \text{Hartman}_6(x_{13:18})$.

For the Hartman6 test function, the covariance parameters of a squared exponential kernel have been estimated once on a subset of points. We give here the obtained length-scales, so that the results can be easily reproduced: $(0.262, 0.435, 0.423, 0.348, 0.314, 0.299)$. For the Hartman18 test function, we use two different sets of covariance parameters: the first one is slightly misspecified since it is

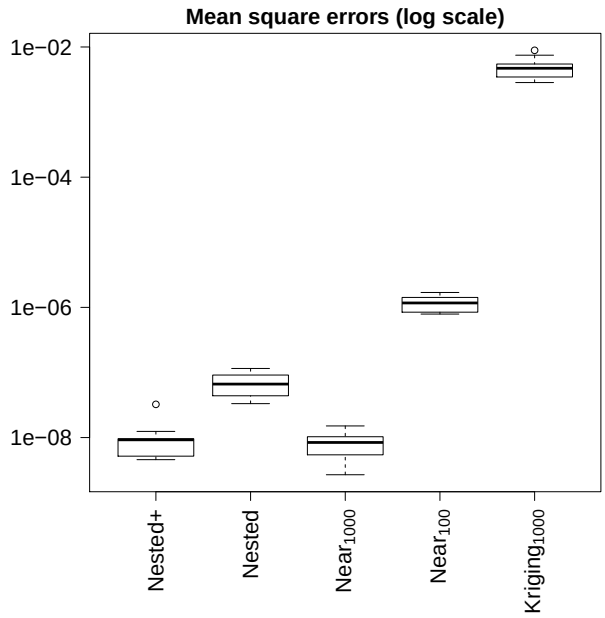
given by the one estimated for Hartman6 repeated three times, the second one is estimated with usual MLE on a subset of 2000 points. Although the model could be improved with a refined estimation of the length-scales, this is sufficient to compare the different methods. The variance parameter has no influence on this comparison since we only consider here the performance of the mean predictors.

We consider in this example $n = 1,000,000$ design points and $q = 100$ predictions points. Several methods are considered:

- the *Kriging* predictor refers to a simple Kriging predictor based on a random sample (without replacement) of 1000 points taken among the initial points. It is mainly computed in order to give an order of the reachable error magnitude for a reasonable learning of the test function, and to see if refined methods really improve the performance of the prediction.
- The *Neighbor* predictor refers to a simple kriging predictor which gives, for each prediction point, the prediction based on its nearest neighbors in the design matrix X. $Near_{100}$ refers to a predictor based on 100 nearest neighbors, $Near_{1000}$ refers to a predictor based on 1000 nearest neighbors.
- The *Nested* method refers to the proposed method in this paper. For one million points, we have chosen a tree structure corresponding to $N = 1000$ groups of points, each group being obtained using *kmeans* clustering algorithm. Two variants are considered: *Nested* refers to a clustering that is built directly without considering the locations of the prediction points. *Nested+* refers to a clustering that is built using these locations (i.e. first $q = 100$ clusters are built around each prediction points, without overlapping, and $N - q = 900$ clusters are built on residual design points). In all cases, depending on the location of design points, each cluster size typically vary between 800 and 1200.

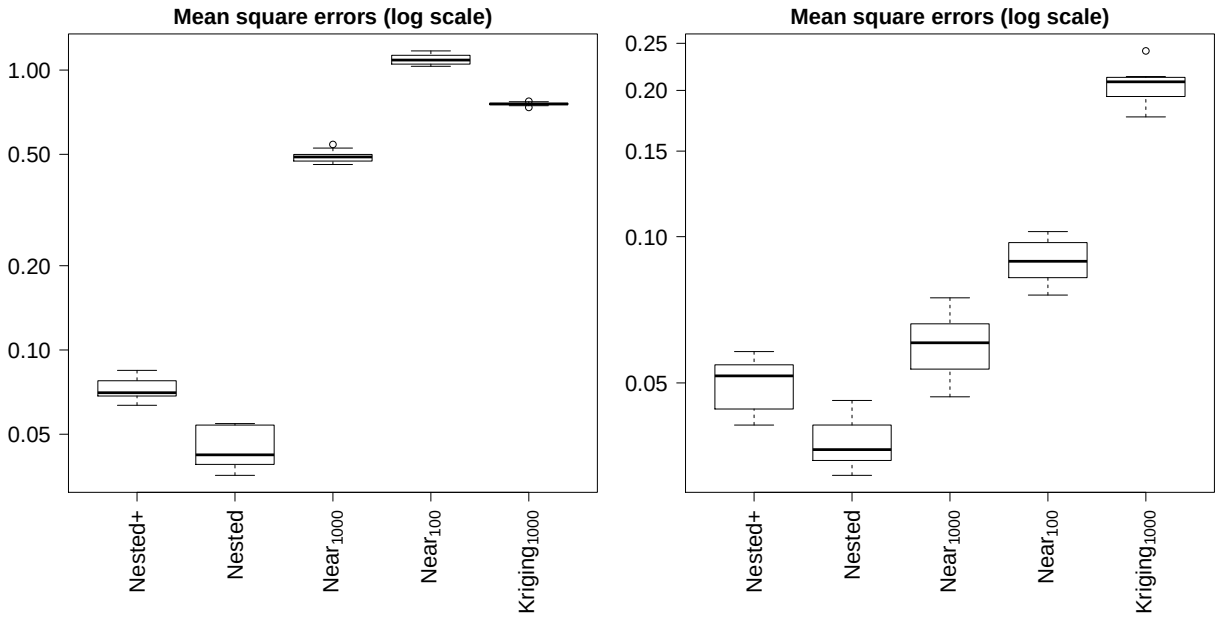
For each run, we draw uniformly a new design matrix X, a new vector of predictions points x, and we analyze the performance of the predictors. To this aim, the predictions are compared to the true chosen test function, and we collect for each run one mean of errors over all prediction points. We reproduce the whole experiment on 10 runs. The results are gathered in the boxplots of Figure 8 and Figure 9.

As announced, the Kriging predictor based on a random sample of 1000 points is mainly given to get an order of the errors magnitude with a reasonable learning of the function, but clearly its complexity is lower, and it does not reach the precision level of other competitors: other methods outperform results based on a random sample of experiments.
The nearest neighbor is a clear improvement over the randomly selected training points and the computation scheme is relatively simple. However it generates non-continuous mean and variance predictions, so the practical interest of such a model (e.g. to perform optimization) may be limited. While it performs very well in small dimension, one can see on Figure 9 that it becomes less attractive in higher dimension since it would require to further increase the number of neighbors to provide competitive predictions. One can also see that nearest neighbors are also quite sensitive to the estimation of the parameters.
The nested procedure has a greater complexity: each run takes around 50 minutes on a modern computer with 16 threads ($1,000,000$ observations, 100 prediction points, Hartman18 test function, 1000 groups and possible multithreading scalability improvements). The execution time for $100,000$ observations and unchanged other settings is around 30 seconds. Our procedure remains here accurate with concatenated parameters, contrarily to other methods. It leads to a very good accuracy and some theoretical advantages previously presented. In relatively small dimension, as in Figure 8, depending on the length-scales of the underlying process, the closest neighbors may be sufficient to explain the local shape of the response. In this case, a judicious choice of the tree structure may improve the accuracy of the nested method, which is comparable to the one of the 1000 closest neighbors. In larger dimension, when local information is not sufficient, the choice of the tree structure has a lower impact, and the refinements of the nested method make sense, as they lead to a better accuracy than



(a) Hartman 6

Figure 8: Boxplot of the mean square errors (in log scale), for the Nested procedure and its variant, the nearest neighbors procedure based on 1000 neighbors or 100 neighbors, and for the Kriging method based on a random sample of 1000 points. We used one million input points, one hundred prediction points and Hartman6 test function.



(a) Hartman 18, concatenated parameters

(b) Hartman 18, estimated parameters

Figure 9: Boxplots of mean square errors (in log scale), for the Nested procedure and its variant, the nearest neighbors procedure based on 1000 neighbors or 100 neighbors, and for the Kriging method based on a random sample of 1000 points. We used one million input points, one hundred prediction points, Hartman18 test function and either concatenated parameters from Hartman6 (left panel) or estimated parameters (right panel).

considered local neighbors methods.

Finally, despite greater complexity, the proposed method is still tractable with one million of observations. It leads to a better accuracy, especially in high dimension. In small dimension and when possible, it can be useful to build the tree structure by using the location of the prediction points, to take the best of both closest neighbors and nested methods. At last, the proposed nested method makes an intensive use of cross-covariances between groups and can surely be improved by using a better estimation of these parameters, or by a transformation of the inputs or the outputs that would make the assumptions more reliable.

5.4 Application to an industrial case study

We consider in this section experimental data on the behavior of a steel test piece subject to cycles of tension-compression. During these cycles, the evolution of the tensile strain in the test piece is monitored over time using two methods: by performing the actual physical experiment and by a numerical simulator based on a Chaboche constitutive equation [Lemaitre and Chaboche, 1994]. The quantity of interest is the misfit between these two experiments. A test piece is described by 6 scalar variables $(E, C_1, C_2, \gamma_1^0, \gamma_2^0, r)$, where E is a logarithm transform of the Young's modulus, C_1, C_2, γ_1^0 and γ_2^0 are parameters related to the kinematic hardening and r is the radius of the plastic surface at the stabilized state. The set of admissible inputs is denoted by $D \subset \mathbb{R}^6$.

Hereafter, we focus on modeling the function $f : D \to \mathbb{R}$ that returns the logarithm of the L^2 norm of the difference between the curve from the actual experiment and the one from the simulator.

In total, we have at our disposal a set of $10,000$ observations $[X, f(X)]$, from which we randomly extract a learning set $[X_l, f(X_l)]$ of $n = 9000$ observations and assign the $n_t = 1000$ remaining observations to a test set $[X_t, f(X_t)]$.

We compare the predictions of $f(X_t)$ obtained from the SPV, PoE, GPoE1, GPoE2, BCM and RBCM aggregation procedures described in Section 5.1 with our nested aggregation procedure. GPoE1 corresponds to (19) with $\beta_i = \frac{1}{2}[\log(\mathrm{V}[Y(x)]) - \log(v_i(x))]$ [Cao and Fleet, 2014] and GPoE2 corresponds to (19) with $\beta_i = 1/p$ [Deisenroth and Ng, 2015]. For all these methods, we consider an aggregation tree of height $\bar{\nu} = 2$ (once sub-models have been evaluated at layer 1, they are all directly aggregated into one value at layer 2), so that p Gaussian process models are directly aggregated. The p subsamples form a partition of $[X_l, f(X_l)]$, which is obtained using the k-means clustering algorithm.

Three covariance functions have been considered for the sub-models: (tensorized) exponential, Matérn 3/2 and Matérn 5/2 (see [Williams and Rasmussen, 2006, Roustant et al., 2012] for the definition of these functions). For all studied methods, the Matérn 5/2 covariance seemed to be the most appropriate to the problem at hand since we obtained overall more accurate results. The results presented hereafter thus focus on this Matérn 5/2 covariance family. Its parameters are estimated with two different techniques depending on the aggregation method: for the methods from the literature and SPV, we follow the recommended procedure which consists in maximizing the sum of the log likelihoods over the p subsamples of $[X_l, f(X_l)]$ (see [Deisenroth and Ng, 2015]). For the proposed nested aggregation, we carry out the stochastic-gradient based estimation method described in Section 4, with starting points set to the maximizer of the sum of the log likelihoods.

To assess the quality of a model with predicted mean m and variance v, we compute three quality criteria using the test set: MSE and MNLP as per Eq. 23 which are small for a good model, and the mean normalized square error (MNSE)

$$MNSE(m, v, f, X_t) = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{(m(x_{t,i}) - f(x_{t,i}))^2}{v(x_{t,i})},$$

which should be close to 1.

	SPV	PoE	GPoE1	GPoE2	BCM	RBCM	Nested
MSE	0.00416	0.0662	0.0033	0.0662	0.604	0.0625	**0.00321**
MNSE	1.27	20.00	4.55	**1.00**	219	60.8	0.846
MNLP	−1.86	7.25	−0.949	−0.765	107	27.2	**−1.97**

Table 1: Prediction performances of the aggregation of $p = 20$ sub-models for the steel piece constraints cycles data set. The investigated prediction performance criteria are the mean square error (MSE) which should be minimal, mean normalized square error (MNSE) which should be close to 1 and mean negative log probability (MNLP) which should be small. Bold figures indicate each line's best performing aggregation method.

	SPV	PoE	GPoE1	GPoE2	BCM	RBCM	Nested
MSE	0.00556	0.811	0.0244	0.811	1.84	0.121	**0.00418**
MNSE	1.20	465	34.2	5.16	980	148	**0.84700**
MNLP	−1.55	230	14.1	2.13	487	71	**−1.7**

Table 2: Prediction performances of the aggregation of $p = 90$ sub-models for the steel piece constraints cycles data set. All other settings are the same as in Table 1.

The prediction results for a given learning and training test set are given in Table 1 for the aggregation of $p = 20$ sub-models and in Table 2 for $p = 90$. It can be seen that in both cases the proposed method outperforms the other aggregation methods for the MSE and MNLP quality criteria. The MSE has the same order of magnitude for the SPV and our aggregation method, where the prediction errors are small compared to the empirical variance of the test outputs $f(x_{t,i})$, $i = 1, ..., n_t$, which is approximately equal to 0.81. In contrast, the MSE can be significantly larger for all the other aggregation procedures. For the PoE, GPoE1, BCM and RBCM aggregation techniques, the values of MNSE are orders of magnitude greater than the target value one, which indicates that the aggregated models are highly overconfident. The GPoE2 aggregation technique is also overconfident when $p = 90$, where its MNSE is equal to 5.16. The SPV and our aggregation methods provide appropriate predictive variances, and our method provides the best combination of predictions and predictive variances, according to the MNLP criterion.

Tables 1 and 2 also show that aggregating $p = 20$ sub-models gives more accurate models than aggregating $p = 90$ sub-models. This suggests that it is a good practice to aggregate few sub-models based on many points instead of aggregating many sub-models based on few points. Although this would require further testing to be confirmed, it is not surprising since aggregation methods rely on some independence assumptions that are not often met in practice.

Tables 3 and 4 show the values of the quality criteria when the subsamples used for the $p = 20$ or $p = 90$ sub-models are randomly generated into the learning set. They can thus be compared to Tables 1 and 2 to study the influence of the choice of the support points of the sub-models: the criteria values are overall better in Tables 1 and 2 so using k-means is beneficial for the aggregation procedures. In addition, our proposed aggregation technique becomes better in comparison to the other methods, and specifically to SPV, when the subsamples are randomly generated.

All previous results have been obtained for a given random choice of the learning and test sets. We now replicate the procedure 20 times, with the same settings as in Tables 1 ($p = 20$; subsamples obtained from the k-means algorithm; Matérn 5/2 covariance function) and 4 ($p = 90$; subsamples randomly selected; Matérn 5/2 covariance function), but with different learning and test sets for each replication. The covariance parameters are reestimated for each learning set, by minimizing the sum of log likelihoods for the SPV, PoE, GPoE1, GPoE2, BCM and RBCM aggregation techniques, and with the proposed leave-one-out estimation procedure for our nested aggregation method. The boxplots of

	SPV	PoE	GPoE1	GPoE2	BCM	RBCM	Nested
MSE	0.0086	0.00763	0.00704	0.00763	0.338	0.274	**0.00539**
MNSE	1.21	9.38	16.6	0.469	178	268	**0.864**
MNLP	−1.25	1.75	5.03	−1.21	86.2	130	**−1.5**

Table 3: Same settings as in Table 1 but when the subsamples are randomly selected.

	SPV	PoE	GPoE1	GPoE2	BCM	RBCM	Nested
MSE	0.0182	0.0293	0.0246	0.0293	0.977	0.686	**0.00575**
MNSE	1.29	42.5	57.2	0.473	852	988	**0.867**
MNLP	−0.804	18.3	25.3	−0.517	423	491	**−1.37**

Table 4: Same settings as in Table 1 but with $p = 90$ sub-models and where the subsamples are randomly selected.

the corresponding 20 mean square errors and mean negative log probability are reported in Figures 10 and 11. These replications confirm the results obtained previously on single instances of the learning and test set: the proposed nested aggregation and covariance parameter estimation jointly give better prediction both for the predicted mean and variance than current existing aggregation techniques.

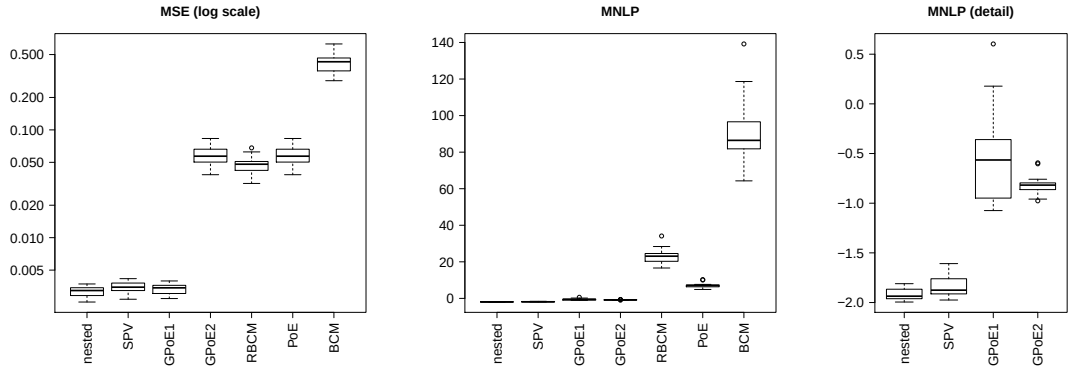


Figure 10: Boxplots of 20 values of the mean square error (MSE) prediction criterion and of the logarithm of the mean negative log probability (MNLP) prediction criterion where the learning and test sets are randomly generated. The settings are as in Table 1 ($p = 20$ subsamples obtained from the k-means algorithm; Matérn 5/2 covariance function). The covariance parameters are estimated by minimizing the sum of log likelihoods for the SPV, PoE, GPoE1, GPoE2, BCM and RBCM aggregation techniques, and with our proposed leave-one-out estimation procedure for the nested aggregation procedure.

Of course, the improvement brought by our proposed aggregation scheme comes with a higher computational cost: the proposed estimation procedure takes a few hours on a personal computer, against a few tens of minutes for the minimization of the sum of the log likelihoods. Similarly, performing 1000 predictions takes around 30 seconds with our proposed optimal aggregation, against around 1 second for the other simpler aggregation procedures. Nevertheless, we believe that the increased accuracy and robustness of the method we propose is worth the additional computational burden in many situations.

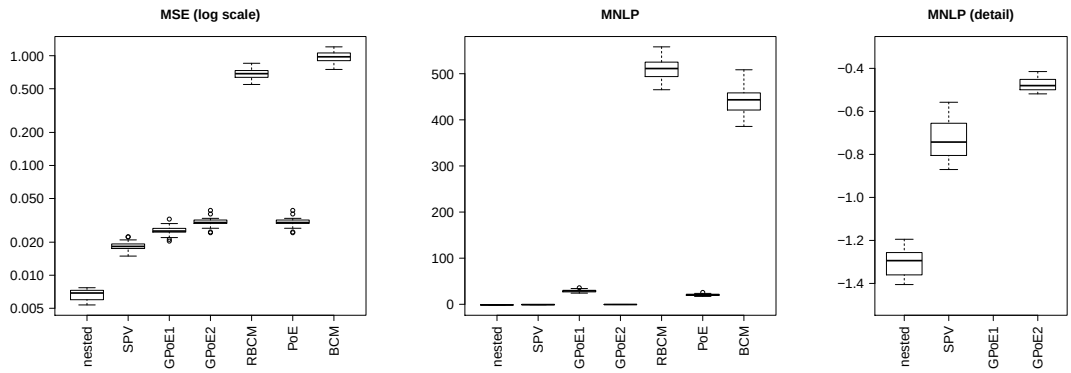


Figure 11: Same settings as in Figure 10 but with $p = 90$ and where the subsamples are randomly selected.

6 Conclusion

We have proposed a new method for aggregating sub-models based on subsets of observation points, with a particular emphasis on Kriging sub-models. Our method can be seen as an optimal linear weighting of sub-models, where the obtained weights are taking into account all pairwise covariances between the sub-models, thus avoiding some usual independence assumptions.

Compared to current existing aggregation techniques, we find several benefits to our our aggregation procedure. First, it has some good theoretical properties, like consistency or optimality based on a slightly different process which can be simulated. We refer again to [Bachoc et al., 2017] for details. Second, a dedicated covariance parameter estimation procedure is provided, based on a gradient descent minimization of leave-one-out cross validation errors, where the predictions are performed using the proposed nested aggregation. Some user-friendly code for computing both prediction and covariance parameter estimation is publicly available.

At last, numerical results are encouraging. In both simulated data and industrial application, our method is shown to outperform state-of-the-art aggregation techniques. This improvement comes with an increased computational cost compared to more basic aggregation methods, but the proposed nested aggregation remains applicable up to $n = 10^6$ observation points, while exact Kriging inference becomes intractable around $n = 10\,000$.

We would like to mention two avenues for future research. First, we show that the aggregation method we propose can be applied recursively, yielding a nested aggregation technique with smaller computational cost. It would be interesting to quantify the practical gain one could obtain on real data sets from this recursive aggregation. Second, we find that the stochastic gradient algorithm we propose could be further investigated. In particular, theoretical properties could be derived, the practical implementation could be improved, and the principle could be extended to other criteria for covariance parameter estimation.

Acknowledgements

Part of this research was conducted within the frame of the Chair in Applied Mathematics OQUAIDO, gathering partners in technological research (BRGM, CEA, IFPEN, IRSN, Safran, Storengy) and academia (Ecole Centrale de Lyon, Mines Saint-Etienne, University of Grenoble, University of Nice, University of Toulouse and CNRS) around advanced methods for Computer Experiments. The authors would like to warmly thank Dr. Géraud Blatman and EDF R&D for providing us the industrial test case. They also thank both editor and reviewers for very precise and constructive comments on this paper. This paper has been finished during a stay of D. Rullière at Vietnam Institute for Advanced

Study in Mathematics, the latter author thanks the VIASM institute and DAMI research chair (Data Analytics & Models for Insurance) for their support.

References

[Bachoc, 2013] Bachoc, F. (2013). Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model mispecification. *Computational Statistics and Data Analysis*, 66:55–69.

[Bachoc et al., 2017] Bachoc, F., Durrande, N., Rullière, D., and Chevalier, C. (2017). Some properties of nested kriging predictors. Technical report hal-01561747.

[Bhatnagar et al., 2013] Bhatnagar, S., Prasad, H., and Prashanth, L. (2013). *Stochastic recursive algorithms for optimization*, volume 434. New York: Springer.

[Cao and Fleet, 2014] Cao, Y. and Fleet, D. J. (2014). Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. *arXiv preprint arXiv:1410.7827v2, CoRR*, abs/1410.7827:1–5. Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS, Montreal.

[Deisenroth and Ng, 2015] Deisenroth, M. P. and Ng, J. W. (2015). Distributed Gaussian processes. *Proceedings of the 32nd International Conference on Machine Learning, Lille, France. JMLR: W&CP volume 37.*

[Genest and Zidek, 1986] Genest, C. and Zidek, J. V. (1986). Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–135.

[Golub and Van Loan, 2012] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

[Guhaniyogi et al., 2011] Guhaniyogi, R., Finley, A. O., Banerjee, S., and Gelfand, A. E. (2011). Adaptive gaussian predictive process models for large spatial datasets. *Environmetrics*, 22(8):997–1007.

[Hensman et al., 2013] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). *Gaussian Processes for Big Data*. Uncertainty in Artificial Intelligence conference. paper Id 244.

[Hinton, 2002] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[Katzfuss, 2013] Katzfuss, M. (2013). Bayesian nonstationary spatial modeling for very large datasets. *Environmetrics*, 24(3):189–200.

[Lemaitre and Chaboche, 1994] Lemaitre, J. and Chaboche, J.-L. (1994). *Mechanics of solid materials*. Cambridge university press.

[Maurya, 2016] Maurya, A. (2016). A well-conditioned and sparse estimation of covariance and inverse covariance matrices using a joint penalty. *The Journal of Machine Learning Research*, 17(1):4457–4484.

[Nickson et al., 2015] Nickson, T., Gunter, T., Lloyd, C., Osborne, M. A., and Roberts, S. (2015). Blitzkriging: Kronecker-structured stochastic Gaussian processes. *arXiv preprint arXiv:1510.07965v2*, pages 1–13.

[Ranjan and Gneiting, 2010] Ranjan, R. and Gneiting, T. (2010). Combining probability forecasts. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(1):71–91.

[Roustant et al., 2012] Roustant, O., Ginsbourger, D., and Deville, Y. (2012). DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1).

[Rue and Held, 2005] Rue, H. and Held, L. (2005). *Gaussian Markov random fields, Theory and applications*. Chapman & Hall.

[Samo and Roberts, 2016] Samo, Y.-L. K. and Roberts, S. J. (2016). String and membrane gaussian processes. *Journal of Machine Learning Research*, 17(131):1–87.

[Santner et al., 2013] Santner, T. J., Williams, B. J., and Notz, W. I. (2013). *The design and analysis of computer experiments*. Springer Science & Business Media.

[Satopää et al., 2016] Satopää, V. A., Pemantle, R., and Ungar, L. H. (2016). Modeling probability forecasts via information diversity. *Journal of the American Statistical Association*, 111(516):1623–1633.

[Scott et al., 2016] Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and big data: The consensus monte carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88.

[Stein, 2012] Stein, M. L. (2012). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

[Stein, 2014] Stein, M. L. (2014). Limitations on low rank approximations for covariance matrices of spatial data. *Spatial Statistics*, 8:1–19.

[Tresp, 2000] Tresp, V. (2000). A bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

[Tzeng et al., 2005] Tzeng, S., Huang, H.-C., and Cressie, N. (2005). A fast, optimal spatial-prediction method for massive datasets. *Journal of the American Statistical Association*, 100(472):1343–1357.

[van Stein et al., 2015] van Stein, B., Wang, H., Kowalczyk, W., Bäck, T., and Emmerich, M. (2015). Optimally weighted cluster kriging for big data regression. In *International Symposium on Intelligent Data Analysis*, pages 310–321. Springer.

[Wahba, 1990] Wahba, G. (1990). *Spline models for observational data*, volume 59. SIAM.

[Wei et al., 2015] Wei, H., Du, Y., Liang, F., Zhou, C., Liu, Z., Yi, J., Xu, K., and Wu, D. (2015). A k-d tree-based algorithm to parallelize kriging interpolation of big spatial data. *GIScience & Remote Sensing*, 52(1):40–57.

[Williams and Rasmussen, 2006] Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[Winkler, 1968] Winkler, R. L. (1968). The consensus of subjective probability distributions. *Management Science*, 15(2):B–61.

[Winkler, 1981] Winkler, R. L. (1981). Combining probability distributions from dependent information sources. *Management Science*, 27(4):479–488.

[Zhang et al., 2015] Zhang, B., Sang, H., and Huang, J. Z. (2015). Full-scale approximations of spatio-temporal covariance models for large datasets. *Statistica Sinica*, pages 99–114.

A Proof of Proposition 4

Complexities: under chosen assumption on α and β coefficients, for a regular tree and in the case of simple Kriging sub-models, $\mathcal{C}_\alpha = \sum_{\nu=1}^{\bar\nu}\sum_{i=1}^{n_\nu}\alpha c_\nu^3 = \alpha\sum_{\nu=1}^{\bar\nu} c_\nu^3 n_\nu$ and $\mathcal{C}_\beta = \sum_{\nu=1}^{\bar\nu}\sum_{i=2}^{n_\nu}\sum_{j=1}^{i-1}\beta c_\nu^2 = \frac{\beta}{2}\sum_{\nu=1}^{\bar\nu} n_\nu(n_\nu-1)c_\nu^2$. Notice that the sum starts from $\nu=1$ in order to include sub-models calculation. *Equilibrated trees complexities*: In a constant child number setting, when $c_\nu = c$ for all ν, the tree structure ensures that $n_\nu = n/c^\nu$, thus as $c = n^{1/\bar\nu}$, we get when $n\to+\infty$, $\mathcal{C}_\alpha \sim \alpha n^{1+\frac{2}{\bar\nu}}$ and $\mathcal{C}_\beta \sim \frac{\beta}{2}n^2$. The result for equilibrated two-layer tree where $\bar\nu = 2$ directly derives from this one, and in this case $\mathcal{C}_\alpha \sim \alpha n^2$ and $\mathcal{C}_\beta \sim \frac{\beta}{2}n^2$ (it derives also from the expressions of $\mathcal{C}_\alpha$, $\mathcal{C}_\beta$, when $c_1 = c_2 = \sqrt{n}$, $n_1 = \sqrt{n}$, $n_2 = 1$). *Optimal tree complexities*: One easily shows that under the chosen assumptions $\mathcal{C}_\beta \sim \frac{\beta}{2}n^2$. Thus, it is indeed not possible to reduce the whole complexity to orders lower than $O(n^2)$. However, one can choose the tree structure in order to reduce the complexity $\mathcal{C}_\alpha$. For a regular tree, $n_\nu = n/(c_1\cdots c_\nu)$ such that $\frac{\partial}{\partial c_k}n_\nu = -\mathbf{1}_{\{\nu\geq k\}}n_\nu/c_k$. Using a Lagrange multiplier ℓ, one defines $\xi(k) = c_k\frac{\partial}{\partial c_k}\left(\mathcal{C}_\alpha - \ell(c_1\cdots c_{\bar\nu} - n)\right) = 3\alpha c_k^3 n_k - \alpha\sum_{\nu=k}^{\bar\nu} c_\nu^3 n_\nu - \ell c_1\cdots c_{\bar\nu}$. The tree structure that minimizes $\mathcal{C}_\alpha$ is such that for all $k<\bar\nu$, $\xi(k) = \xi(k+1) = 0$. Using $c_{k+1}n_{k+1} = n_k$, one gets $3c_{k+1}^2 = 2c_k^3$ for all $k<\bar\nu$, and setting $c_1\cdots c_{\bar\nu} = n$, $c_\nu = \delta\left(\delta^{-\bar\nu}n\right)^{\frac{\delta^{\nu-1}}{2(\delta^{\bar\nu}-1)}}$, $\nu = 1,\ldots,\bar\nu$, with $\delta = \frac{3}{2}$. Setting $\gamma = \frac{27}{4}\delta^{-\frac{\bar\nu}{\delta^{\bar\nu}-1}}\left(1-\delta^{-\bar\nu}\right)$. After some direct calculations this tree structure corresponds to complexities, $\mathcal{C}_\alpha = \gamma\alpha n^{1+\frac{1}{\delta^{\bar\nu}-1}}$ and $\mathcal{C}_\beta \sim \frac{\beta}{2}n^2$. In a two-layers setting one gets $c_1 = \left(\frac{3}{2}\right)^{1/5}n^{2/5}$ and $c_2 = \left(\frac{3}{2}\right)^{-1/5}n^{3/5}$, which leads to $\mathcal{C}_\alpha = \gamma\alpha n^{9/5}$ and $\mathcal{C}_\beta = \frac{\beta}{2}n^2 - \frac{\beta}{2}\left(\frac{3}{2}\right)^{\frac{1}{5}}n^{\frac{7}{5}}$, where $\gamma = (\frac{2}{3})^{-2/5} + (\frac{2}{3})^{3/5} \simeq 1.96$ (eventually notice that even for values of n of order 10^5, terms of order like $n^{9/5}$ are not necessarily negligible compared to those of order n^2, and that $\mathcal{C}_\beta$ is slightly affected by the choice of the tree structure, but the global complexity benefits from the optimization of $\mathcal{C}_\alpha$). *Storage footprint*: First, covariances can be stored in triangular matrices. So temporary objects M, k and K in Algorithm 1 require the storage of $c_{\max}(c_{\max}+5)/2$ real values. For a given step ν, $\nu\geq 2$, building all vectors α_i requires the storage of $\sum_{i=1}^{n_\nu} c_i^\nu = n_{\nu-1}$ values. At last, for a given step ν, we simultaneously need objects $M_{\nu-1}, K_{\nu-1}, M_\nu, K_\nu$, which require the storage of $n_{\nu-1}(n_{\nu-1}+3)/2 + n_\nu(n_\nu+3)/2$ real values. In a regular tree, as n_ν is decreasing in ν, the storage footprint is $\mathcal{S} = (c_{\max}(c_{\max}+5) + n_1(n_1+5) + n_2(n_2+3))/2$. Hence the equivalents for $\mathcal{S}$ for the different tree structures, $\mathcal{S}\sim n$ for the two-layer equilibrated tree, $\mathcal{S}\sim\frac{1}{2}n^{2-2/\bar\nu}$ for the $\bar\nu$-layer, $\bar\nu>2$, and the indicated result for the optimal tree. Simple orders are given in the proposition, which avoids separating the case $\bar\nu = 2$ and a cumbersome constant for the optimal tree.