

A reproducible comparative study of categorical kernels for Gaussian process regression, with new clustering-based nested kernels

Raphaël CARPINTERO PEREZ^{a,b}, Sébastien DA VEIGA^c, Josselin GARNIER^b

^a*Safran Tech, Digital Sciences & Technologies, 78114 Magny-Les-Hameaux, France*

^b*Centre de Mathématiques Appliquées, Ecole polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France*

^c*Univ Rennes, Ensai, CNRS, CREST - UMR 9194, F-35000 Rennes, France*

Abstract

Designing categorical kernels is a major challenge for Gaussian process regression with continuous and categorical inputs. Despite previous studies, it is difficult to identify a preferred method, either because the evaluation metrics, the optimization procedure, or the datasets change depending on the study. In particular, reproducible code is rarely available. The aim of this paper is to provide a reproducible comparative study of all existing categorical kernels on many of the test cases investigated so far. We also propose new evaluation metrics inspired by the optimization community, which provide quantitative rankings of the methods across several tasks. From our results on datasets which exhibit a group structure on the levels of categorical inputs, it appears that nested kernels methods clearly outperform all competitors. When the group structure is unknown or when there is no prior knowledge of such a structure, we propose a new clustering-based strategy using target encodings of categorical variables. We show that on a large panel of datasets, which do not necessarily have a known group structure, this estimation strategy still outperforms other approaches while maintaining low computational cost.

Keywords: Gaussian process regression, categorical data, group kernels, clustering, benchmark

1. Introduction

Computer experiments have become an invaluable tool in many engineering and industrial applications due to their ability to approximate expensive physical experiments or computational models (Santner et al., 2003). The study of complex systems can involve different objectives, ranging from sensitivity analysis (Da Veiga et al., 2021) to design of experiments (Gramacy, 2020), uncertainty propagation (Girard, 2004), calibration (Kennedy and O’Hagan, 2001) and black-box optimization (Jones et al., 1998). In this context, a key objective is to create a cheap-to-evaluate surrogate model from a true underlying function while providing good predictivity and characterizing sensible predictive uncertainty. To achieve this goal, one often resorts to Gaussian processes (Rasmussen, 2003) also called Kriging models (Krig, 1951), known for their ability to generalize well even when the number of observations is small. The vast majority of papers dealing with Gaussian process regression are restricted to the case where the inputs are continuous, with universally known kernels that can be used off-the-shelf. In practice, however, it is common to have access to inputs in more complicated forms, such as graphs (Carpintero Perez et al., 2024), meshes (Kabalan et al., 2025), images (Yin and Du, 2022) or even probability distributions (Meunier et al., 2022), which all require specific kernels to be defined. In this study, we specifically address the case of mixed continuous and categorical inputs. This problem is not new, and myriad of applications can be found in fields such as chemical and antibody design (Gómez-Bombarelli et al., 2018; Khan et al., 2023), aircraft design (Beauthier et al., 2014; Jesus et al., 2021; Bartoli et al., 2019; Saves et al., 2022), neural architecture search (Wistuba et al., 2019), vegetative filter strips Lauvernet and Helbert (2020) and even cooking (Kochanski et al., 2017).

The term categorical is used in different senses in the literature. A first distinction must be made between categorical variables, otherwise known as nominal, and ordinal variables, which can take a finite number of ordered values. Some approaches focus specifically on binary variables, used for instance in boolean feasibility problems, while others handle categorical variables when they form strings (Moss et al., 2020; Grosnit et al., 2022). It should also be noted that some papers deal with overly specific cases, such as category-specific continuous variables (Gopakumar

et al., 2018; Nguyen et al., 2020). Finally, one of the most common applications of mixed-variable Gaussian processes is Bayesian optimization: the focus is therefore on optimization itself, sometimes neglecting the kernel’s predictive performance. In such setting the kernel is often under-parameterized or simply ignores the categorical nature of inputs.

Although some reviews exist (Zhang and Notz, 2015; Pelamatti, 2020; Roustant et al., 2020; Saves, 2024) and numerous approaches have been proposed, we argue that it is difficult to make a clear recommendation on the categorical kernels to be used in practice. Indeed, papers are very rarely accompanied by codes, and reproducibility of some results is often cumbersome. In addition, optimization of kernel parameters is very sensitive to implementation details, from the choice of the optimizer, the number of multi-start trials or the bounds. Without available code, trust in previous benchmarks may be problematic. Our aim here is to close this gap by providing the practitioner a guide to choosing a categorical kernel, with reproducible experiments on several tasks. Our contribution is three-fold. First, we provide an overview of recent categorical kernels, with a critical look at some of the kernels used in Bayesian optimization. In particular, we point out that several kernels actually boil down to a naive one-hot encoding transformation. Second, all these kernels are implemented in the accompanying code, with optimization strategies clearly stated and finally compared on common datasets, with two optimization settings: one where we use the default values of optimization control parameters (as in most Gaussian process software), and one where we consider much less restrictive values (e.g. we allow for larger numbers of iterations) in order to increase the chance to reach convergence. This provides a ranking of the methods, which is used to identify the best performing methods in an objective way, while also considering their computational cost. In the case of multiple categorical variables, we also evaluate the accuracy when they are considered separately or when product variables are used. Finally, we focus on the identification of possible groups within levels when they are unknown or when no such prior assumption exists, in order to be able to use nested kernels (Roustant et al., 2020) in a general context. In the latter, it is suggested that these groups may be selected by clustering the covariance matrices obtained after running another method in a preliminary step. However, numerical experiments were not carried out to explore this idea, and the coupling with another supervised approach may be a computational bottleneck. We propose instead to use a procedure that does not require any prior training. This new procedure represents each level thanks to the distribution of the output conditioned by the categorical variable taking that specific level, thus making it possible to define a similarity between the levels informed by the output variable, followed by a clustering step. We show that this approach outperforms other methods while maintaining low computational cost. To summarize, our main objectives are the following:

- Present a critical review of the most popular strategies and their variants, including group kernels, hyperspheres and latent encodings
- Propose an extensive benchmark in Python with reproducible results, which accounts for the computational time
- Investigate different initialization strategies for the groups in nested kernels when groups are unknown

The article is organized as follows. We start by introducing Gaussian process regression in the special case of mixed-variable inputs and review classical categorical kernels in Section 2. We then divide our experiences into two distinct parts. Section 3 focuses on datasets with known groups, while Section 4 highlights datasets with no known groups and discusses strategies for identifying groups in this case.

2. Categorical variables in Gaussian process regression

2.1. Notations and overview

Let us consider the case where we observe both $n \in \mathbb{N}$ continuous variables with values in $\mathbb{R}$, and m categorical variables with respectively $C_1, \dots, C_m$ levels (also called categories or modalities). Without loss of generality, we denote the sets of levels $\mathcal{Z}_i = \{1, \dots, C_i\} = \llbracket C_i \rrbracket$ for variable i , so that the space of categorical variables is written as $\mathcal{Z} = \prod_{i=1}^m \mathcal{Z}_i$. The input space is thus $\mathcal{W} = \mathbb{R}^n \times \mathcal{Z}$. We consider the task of learning a function $f : \mathcal{W} \rightarrow \mathbb{R}$ from a dataset $\mathcal{D}$ of N observations $\mathcal{D} = \{(W^{(i)}, y^{(i)})\}_{i=1}^N$ where $W^{(i)} = (x^{(i)}, z^{(i)})$, with noisy observations $y^{(i)} = f(W^{(i)}) + \epsilon^{(i)}$ for $i \in \llbracket N \rrbracket$ at input locations $\mathbf{W} = (W^{(i)})_{i=1}^N$, where $\epsilon^{(i)} \sim \mathcal{N}(0, \eta^2)$ is an i.i.d. Gaussian additive noise. We denote $\mathbf{y} = (y^{(i)})_{i=1}^N$ and $\mathbf{f}_* := (f(W_*^{(i)}))_{i=1}^{N^*}$ the values of f at new test locations $\mathbf{W}_* = (W_*^{(i)})_{i=1}^{N^*}$. A constant-mean Gaussian

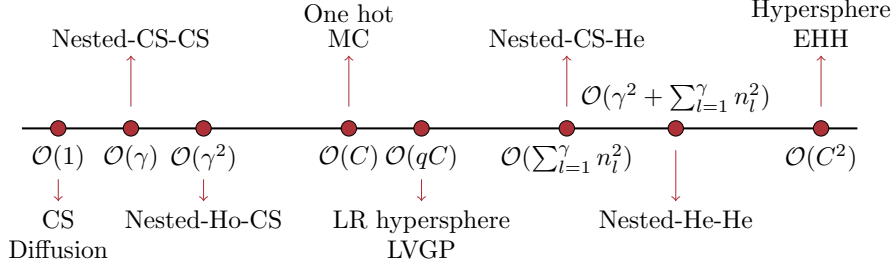


Figure 1: Number of parameters of several categorical kernels. C : number of levels, q : latent dimension (for encoding-based or low-rank approaches), γ : number of groups (for nested kernels), n_l : number of levels per group (for nested kernels). Remark that $C = \sum_{l=1}^{\gamma} n_l$ so that we always have $C \leq \sum_{l=1}^{\gamma} n_l^2$ and $\sum_{l=1}^{\gamma} n_l^2 \leq C^2 \leq \gamma \sum_{l=1}^{\gamma} n_l^2$.

process (GP) prior is placed on the function f . To simplify the notations, we will use a zero-mean here. It follows that the joint distribution of the noisy function values $\mathbf{y}$ at the observed locations and the function values $\mathbf{f}_*$ at the test locations writes (see, *e.g.*, Rasmussen (2003))

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \eta^2 \mathbf{I} & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right), \quad (1)$$

where $\mathbf{K}$, $\mathbf{K}_{**}$, $\mathbf{K}_*$ are the train, test and test/train Gram matrices, respectively.

The posterior distribution of $\mathbf{f}_*$, obtained by conditioning the joint distribution on the observed noisy data, is also Gaussian: $\mathbf{f}_* | \mathbf{W}, \mathbf{y}, \mathbf{W}_* \sim \mathcal{N}(\bar{\mathbf{m}}, \bar{\Sigma})$ with mean and covariance given by

$$\bar{\mathbf{m}} = \mathbf{K}_* (\mathbf{K} + \eta^2 \mathbf{I})^{-1} \mathbf{y}, \quad (2)$$

$$\bar{\Sigma} = \mathbf{K}_{**} - \mathbf{K}_* (\mathbf{K} + \eta^2 \mathbf{I})^{-1} \mathbf{K}_*^\top. \quad (3)$$

The mean of the posterior distribution is used as a predictor, and predictive uncertainties can be obtained through the posterior covariance matrix.

Since prediction formulas are in closed-form, the cornerstone of GP regression lies in the choice of the symmetric kernel function $k : \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{R}$, which must be positive definite (also called valid kernel).

Although some approaches try to build a kernel on $\mathcal{W}$ directly (Oh et al., 2021), the standard way to tackle the problem is based on the construction and combination of separate kernels on the continuous and categorical parts. Suppose we have access to a valid kernel for the continuous variables $k_{\text{cont}} : \mathbb{R}^n \times \mathbb{R}^n$, and one for the categorical variables $k_{\text{cat}} : \mathcal{Z} \times \mathcal{Z}$. Usual valid combinations include the product, sum, or ANOVA.

RBF and Matérn kernels are reference choices when variables are continuous. For categorical variables, it is customary to consider each categorical variable separately, and to combine them after. Although additive kernels proposed by Deng et al. (2017) offer an interesting alternative with the sum, the results of Roustant et al. (2020) show that they have inferior performance, so we limit our study to the most common product kernels. We thus seek to define a categorical kernel $k_{\text{cat}}(z, z') = \prod_{i=1}^m k_{\text{cat},i}(z_i, z'_i)$ where $z = (z_1, \dots, z_m) \in \mathcal{Z}$, $z' = (z'_1, \dots, z'_m) \in \mathcal{Z}$. In the next section, in order to simplify notations, we consider that $m = 1$, $\mathcal{Z} = \llbracket C \rrbracket$ and $k_{\text{cat}} : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$. We identify two ways of constructing such kernels. The first one is to create a data representation, or encoding, which is then plugged into a traditional continuous kernel. The second, as described by Zhou et al. (2011), relies on the fact that a categorical variable is characterized by a finite number of levels and thus can be described by a finite number of covariance values that we denote by $\mathbf{T} = (T_{z,z'}) \in \mathbb{R}^{C \times C}$, such that $k_{\text{cat}}(z, z') = T_{z,z'}$. By choosing a suitable parameterization of the covariance matrix, the positive definiteness of k_{cat} follows from the positive definiteness of $\mathbf{T}$. Each method has its own number of parameters to be inferred from the data, see Figure 1 for a summary of the main approaches described in the following subsections.

2.2. Encoding

One-hot encoding.

A categorical variable with C levels can be represented by a vector of size C composed of zeros and ones. $z \in \llbracket C \rrbracket$ can be written as $E(z) \in \{0, 1\}^C$ where $E(z)_i = \mathbb{1}_{\{i=z\}}$ for $i = 1, \dots, C$. The encoded variable is then plugged into continuous kernels, such as the RBF: $k_{\text{cat}}(z, z') = \prod_{i=1}^C e^{-\frac{(E(z)_i - E(z')_i)^2}{2\theta_i^2}}$ where $\theta = (\theta_1, \dots, \theta_C) \in (0, +\infty)^C$. Relaxation approaches are often used in Bayesian optimization (Garrido-Merchán and Hernández-Lobato, 2020; Karlsson et al., 2020; Halstrup, 2016; Kim et al., 2022), where one-hot encoded variables are further allowed to take values in $[0, 1]^C$. However, note that these kernels have the trivial expression:

$$k_{\text{cat}}(z, z') = \mathbb{1}_{\{z=z'\}} + e^{-\frac{1}{2}(\theta_z^{-2} + \theta_{z'}^{-2})} \mathbb{1}_{\{z \neq z'\}}, \quad (4)$$

which is the sum of a diagonal matrix and a rank-one matrix. This shows the limited representational power of one-hot encoding.

Latent variables.

Instead of using a fixed representation, Latent Variable Gaussian Process (LVGP) (Zhang et al., 2020) proposes to infer a representation in dimension q where $1 \leq q < C$. Indeed, it is common in engineering that categorical variables are explained by unobserved continuous ones (e.g., a material type is characterized by several physical properties). Such a representation is defined by a mapping $\phi : \llbracket C \rrbracket \rightarrow \mathbb{R}^q$, parameterized by $qC - \frac{q(q+1)}{2}$ variables:

$$\begin{aligned} \phi(1) &= (0, \quad 0, \quad \dots & 0) \\ \phi(2) &= (\theta_{2,1}, \quad 0, \quad \dots & 0) \\ \phi(i) &= (\theta_{i,1}, \quad \theta_{i,2}, \quad \dots, \quad \theta_{i,i-1}, \quad 0, \quad \dots & 0) \quad \text{for } 2 \leq i \leq q \\ \phi(i) &= (\theta_{i,1}, \quad \theta_{i,2}, \quad \dots & \theta_{i,q}) \quad \text{for } i > q. \end{aligned} \quad (5)$$

Finally, LVGP relies on a continuous RBF kernel between these low-dimensional embeddings as follows:

$$k_{\text{cat}}(z, z') = \exp(-\|\phi(z) - \phi(z')\|_2^2).$$

According to Zhang et al. (2020), while choosing $q = 1$ lacks expressivity and $q = C - 1$ is too large, using $q = 2$ is sufficient in many practical cases. The LVGP kernel was successfully used in Cuesta Ramirez et al. (2022). In the case where several categorical variables are observed, we can also mention the work of (Oune and Bostanabad, 2021), where they propose a common latent representation for all possible levels in the product space instead of defining a latent representation for each categorical variable separately. Note also that another way to learn latent representations is to use deep generative models (Notin et al., 2021; Deshwal and Doppa, 2021; Maus et al., 2022).

2.3. Covariance matrix parameterization

In the covariance matrix parameterization framework, recall that $k^{\text{cat}}(z, z') = T_{z,z'}$ where $\mathbf{T}$ is the $C \times C$ kernel matrix, with unknown entries to be inferred. Symmetry and positive definiteness properties of $\mathbf{T}$ are usually exploited to find simple parameterizations (Pinheiro and Bates, 1996), but the most popular ones involve spectral or Cholesky decompositions, and sparse representations which we detail below.

Spectral decomposition.

Since $\mathbf{T}$ needs to be symmetric, one can use the spectral decomposition $\mathbf{T} = \mathbf{P}\mathbf{D}\mathbf{P}^\top$, where $\mathbf{P}$ is orthogonal and $\mathbf{D}$ is diagonal. Then, the problem boils down to finding a suitable parameterization of the orthogonal matrix $\mathbf{P}$. These include, for example, the Eulerian angles or the Cayley and the Householder transforms (Khuri and Good, 1989; Shepard et al., 2015). These representations require learning $C(C - 1)/2$ parameters, plus the C parameters for the diagonal matrix parameters. These representations are however less popular than the following ones, since they only exploit symmetry.

Cholesky decomposition.

Rather than the spectral decomposition, it is more common to use the Cholesky decomposition, which applies to symmetric and positive definite matrices (Rapisarda et al., 2007). The Cholesky decomposition writes as $\mathbf{T} = \mathbf{L}\mathbf{L}^\top$ where $\mathbf{L}$ is a lower triangular matrix. As described in Zhou et al. (2011), such lower triangular matrices can be parameterized by the following **heteroscedastic hypersphere** (He) decomposition, also called unrestrictive covariance:

$$\begin{aligned} L_{1,1} &= \theta_{1,0} \\ L_{i,1} &= \theta_{i,0} \cos(\theta_{i,1}) \text{ for } 2 \leq i \leq C \\ L_{i,i} &= \theta_{i,0} \prod_{l=1}^{i-1} \sin(\theta_{i,l}) \text{ for } 2 \leq i \leq C \\ L_{i,j} &= \theta_{i,0} \cos(\theta_{i,j}) \prod_{l=1}^{j-1} \sin(\theta_{i,l}) \text{ for } 2 \leq j < i \leq C \end{aligned} \quad (6)$$

where $\theta_{i,j} \in (0, \pi)$ for $1 \leq j < i \leq C$ and $\theta_{i,0} > 0$ for $1 \leq i \leq C$. The intuition behind this parameterization is that each level can be represented as a point on the surface of the C -dimensional hypersphere. By default, it depends on $\frac{C(C+1)}{2}$ parameters. The number of parameters can be further reduced to $\frac{C(C-1)}{2}$ when we assume that all levels share the same variance, i.e. $\theta_{i,0} = 1$ for all i : this is the **homoscedastic hypersphere** (Ho).

Furthermore, in this setting the entries of $\mathbf{T}$ take values in $(-1, 1)$, thus allowing negative correlations. Restricting $\theta_{i,j} \in (0, \frac{\pi}{2})$ for $1 \leq j < i \leq C$, all entries of $\mathbf{T}$ take value in $(0, 1)$, thus exhibiting only positive correlations.

The hypersphere parameterization (He or Ho, with or without negative correlations) has the advantage of being very rich, but has a very large number of parameters to optimize, which can be prohibitive when the number of levels increases, in particular when using values for the optimizer control parameters that are adapted to such dimensionality. This explains why sparse variants have recently emerged.

Low-rank matrices.

Similarly to the LVGP approach, it is possible to assume that the data live in a lower rank space such that $\mathbf{T} = \mathbf{U}\mathbf{U}^\top$, where $\mathbf{U}$ is a $C \times q$ matrix, and $q < C$. Starting from the hypersphere parameterization, and fixing some angles to 0, Kirchhoff and Kuhnt (2020) build a representation with $(q-1)(C-\frac{q}{2})$ parameters.

Multiplicative covariance.

Another kernel that is present in the literature is the multiplicative covariance kernel (McMillan et al., 1999; Qian et al., 2008). Its form is as follows:

$$T_{z,z'} = \mathbb{1}_{\{z=z'\}} + \exp(-\theta_z - \theta_{z'}) \mathbb{1}_{\{z \neq z'\}}$$

where $\theta_i \in [0, +\infty)$ for $1 \leq i \leq C$. Note the striking similarity with Equation (4): this multiplicative kernel in fact amounts to choosing a one-hot encoding of the variables, and then applying a continuous exponential kernel. As expected, such under-parameterization fails to capture commonly occurring correlation structures when $C \geq 4$ as pointed out by Zhang and Notz (2015) and Zhang et al. (2020).

Parametric exponentiation.

Saves et al. (2023) generalize the multiplicative covariance by introducing a symmetric positive definite matrix $\boldsymbol{\tau} \in \mathbb{R}^{C \times C}$ characterizing the correlations between all the levels of the categorical variable:

$$\mathbf{T}_{z,z'} = \exp(-\tau_{z,z} - \tau_{z',z'} - 2\tau_{z,z'}) \mathbb{1}_{\{z \neq z'\}} + \mathbb{1}_{\{z=z'\}}. \quad (7)$$

$\boldsymbol{\tau}$ is further parameterized using the hypersphere decomposition (6), with parameters $\theta_{i,j} \in (0, \frac{\pi}{2})$ for $1 \leq j < i \leq C$, $\theta_{i,0} = 1$ for $1 \leq i \leq C$ and a small tolerance $0 < \epsilon \ll 1$. This amounts to defining $\tau_{i,i} = \theta_{i,i} \geq 0$ for $1 \leq i \leq C$ and $\tau_{i,j} = \frac{\log(\epsilon)}{2} ((\mathbf{L}\mathbf{L}^\top)_{i,j} - 1)$ for $i \neq j$. With this parameterization, called Fully Exponential (FE), the covariance matrix $\mathbf{T}$ is positive definite, with values in $[\epsilon, 1)$ and uses $\frac{C(C+1)}{2}$ parameters. Saves et al. (2023) also propose an alternative way to build the correlation matrix (7) by imposing that $\theta_{i,i} = 0$ for $1 \leq i \leq C$, thus leading to:

$$T_{z,z'} = \exp(-2\tau_{z,z'}) \mathbb{1}_{\{z \neq z'\}} + \mathbb{1}_{\{z=z'\}}.$$

They call it **exponential homoscedastic hypersphere** (EHH), and it depends on $\frac{C(C-1)}{2}$ parameters.

Compound symmetry.

Unlike hypersphere parameterizations, which have a number of parameters growing quadratically with the number of levels, some models propose to describe the covariance matrices with a number of parameters independent of the number of levels. For example, the simplest models consist in setting a constant covariance value between levels. The **compound symmetry** (CS) model (Katz, 2011), also called exchangeable covariance, writes $\mathbf{T}$ as a function of two values, c for covariance and v for variance between levels:

$$T_{z,z'} = v\mathbb{1}_{\{z=z'\}} + c\mathbb{1}_{\{z \neq z'\}}.$$

This defines a valid kernel as long as $v > 0$ and $\frac{c}{v} \in (-\frac{1}{C-1}, 1)$. Subcases are occasionally found, such as the Aitchison-Aitken kernels (Aitchison and Aitken, 1976; Watanabe, 2023), fixing $v = 1 - b$ and $c = \frac{b}{C-1}$. More surprisingly, we can also mention the kernels of Oh et al. (2019) for m categorical variables. Their approach proposes to represent the combinatorial space thanks to a graph Cartesian product of combinatorial graphs, and then to use a diffusion kernel (Kondor and Lafferty, 2002) on the factor graph. However, without any further knowledge about the graph structure of the combinatorial space, the diffusion kernel for complete graphs subsequently has a closed form expression which does not require any spectral decomposition:

$$T_{z,z'} = \left(\frac{1 + (C-1)e^{-\beta C}}{C} \right) \mathbb{1}_{\{z=z'\}} + \left(\frac{1 - e^{-\beta C}}{C} \right) \mathbb{1}_{\{z \neq z'\}}$$

where $\beta > 0$. Hybrid kernels of Deshwal et al. (2021a) also take a very similar form, while Mercer features of (Deshwal et al., 2021b) also use these diffusion kernels and derive an explicit representation of m binary variables. After rewriting, it is clear that these kernels are equivalent to the simple compound symmetry kernel, although this is never mentioned explicitly in the literature.

Nested kernels.

The **nested kernels**, also called group kernels or generalized compound symmetry (GCS), were proposed by Roustant et al. (2020) as a generalization of the group compound symmetry model of (Qian et al., 2008). In particular, they studied validity conditions of such kernels, which previously had no theoretical guarantees. The main assumption for nested kernels is that the levels can be divided into different groups. They consist in building a sparse representation of $\mathbf{T}$ which involves two types of blocks: within-group covariances and covariances between groups. Suppose $\llbracket C \rrbracket$ is partitioned into γ groups $\mathcal{G}_1, \dots, \mathcal{G}_\gamma$ of respective sizes n_l , $1 \leq l \leq \gamma$. The nested kernel writes as follows:

$$\mathbf{T} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{B}_{1,2} & \cdots & \mathbf{B}_{1,\gamma} \\ \mathbf{B}_{2,1} & \mathbf{W}_2 & \ddots & \mathbf{B}_{2,\gamma} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{B}_{\gamma,1} & \cdots & \mathbf{B}_{\gamma,\gamma-1} & \mathbf{W}_\gamma \end{pmatrix}$$

where the diagonal blocks $\mathbf{W}_l$ are $n_l \times n_l$ matrices containing the within-group covariances and $\mathbf{B}_{l,l'}$ are constant matrices containing between-group covariances. The authors put forward simple conditions to verify the positive definiteness of $\mathbf{T}$: $\mathbf{W}_l$ and $\mathbf{W}_l$ minus its mean need to be positive semidefinite for all l and the $\gamma \times \gamma$ matrix $\tilde{\mathbf{T}}$ obtained by averaging each block of $\mathbf{T}$ needs to be positive definite. In order to meet these conditions, they propose a way to parameterize valid GCS block matrices by a family of covariance matrices of smaller size. By using $\mathbf{B}^*$ a $\gamma \times \gamma$ matrix of between-group means covariance, and $\mathbf{M}_l$ a centered covariance matrix of size $(n_l - 1) \times (n_l - 1)$ for $l = 1, \dots, \gamma$, they form the following block matrix:

$$\begin{aligned} \mathbf{W}_l &= \mathbf{B}_{l,l}^* \mathbf{J}_{n_l, n_l} + \mathbf{A}_l \mathbf{M}_l \mathbf{A}_l^T \\ \mathbf{B}_{l,l'} &= \mathbf{B}_{l,l'}^* \mathbf{J}_{n_l, n_{l'}} \end{aligned}$$

where $\mathbf{J}_{a,b}$ is the $a \times b$ matrix of ones and $\mathbf{A}_l$ is a $n_l \times (n_l - 1)$ matrix whose columns form an orthonormal basis of $1_{n_l}^\perp$. Different levels of sparsity can be obtained with different choices of generator matrices as detailed in Table 1 taken from Roustant et al. (2020).

Table 1: Parameterization details for some valid GCS block covariance matrices (the two first columns give the parametric setting).

$\mathbf{M}_l$	$\mathbf{B}^*$	$\mathbf{W}_l$	$\mathbf{B}_{l,l'}$	Number of parameters
$v_{\mathcal{M}} \mathbf{I}_{n_l-1}$	CS(v, c)	CS(v_l, c_l)	c	$\gamma + 2$
$v_{\mathcal{M}} \mathbf{I}_{n_l-1}$	Ho/He	CS(v_l, c_l)	$c_{l,l'}$	$\frac{\gamma(\gamma+3)}{2}$
Ho/He	CS(v, c)	Ho/He	c	$\sum_{l=1}^{\gamma} \frac{n_l(n_l+1)}{2} + 2$
Ho/He	Ho/He	Ho/He	$c_{l,l'}$	$\sum_{l=1}^{\gamma} \frac{n_l(n_l+1)}{2} + \frac{\gamma(\gamma+1)}{2}$

2.4. Kernels from the Bayesian optimization literature

In the Bayesian optimization community, it is much less common to use kernels such as hyperspheres, while similarity functions (not always positive definite) are often used.

Hamming Embeddings via Dictionaries (HED), see Deshwal et al. (2023), select a number of discrete structures from the categorical space (the dictionary) and use them to define an ordinal embedding for high-dimensional combinatorial structures. Given a dictionary with $q \geq 1$ elements, the Hamming embedding vector of a level corresponds to the vector of Hamming distances between the level and all the elements in the dictionary. However, this approach requires access to a sufficient number of categorical variables with low number of levels to be relevant. Alternatively, for categorical inputs expressed as string data, adapted kernels are also often used (Grosnit et al., 2022; Moss et al., 2020).

In many papers in the Bayesian optimization framework, the Gower distance (Halstrup, 2016; Gower, 1971) is very popular. For m categorical variables, the Gower distance between two inputs simply counts the average number of times the categorical variables are the same. The associated Gower distance kernel plugs the Gower distance into a power exponential kernel, but the kernel is not positive definite. Similar indefinite kernels can be found in Ru et al. (2020) and Wan et al. (2021).

2.5. Available categorical kernels in Python

Finally, from a practitioner perspective, we discuss which categorical kernels mentioned above are easily available in Python. Most Bayesian optimization and GP toolboxes are limited to continuous inputs, like Spearmint (Snoek et al., 2012), GPyTorch (Gardner et al., 2018), GPy (GPy, since 2012), BoTorch (Balandat et al., 2020). The OpenTurns software only features LVGP (Baudin et al., 2015). The MCBO framework of (Drechkowski et al., 2024) only considers under-parameterized kernels based on the Gower distance or the diffusion kernels, HED or kernels for string data. The SMT2 toolbox (Saves et al., 2024) is the most complete as it offers continuous relaxation, Gower distance, homoscedastic hypersphere and exponentiation homoscedastic hypersphere. As a sidenote, the R package kergp (Deville et al., 2024) considers hypersphere kernels and nested kernels. But all in one, it is obvious that the existing implementations only offer a very limited set of choices.

3. Implementation and experiments with known group structure

In this section, we describe the kernels considered in our experiments, and detail our implementation choices. We then describe our experimental protocol, and first evaluate methods on test cases with a known group structure. Experiments without such a structure will be discussed in Section 4. We start by recalling the list of kernels used in all experiments in Table 2, and those used only in the known-group setting in Table 3. The code needed to reproduce the experiments can be found at the following URL: https://gitlab.com/drti/cat_gp/.

3.1. Implementation details

The experimental setting we describe here for Section 3 is the same as for Section 4. Important details about the experiments, in particular the parameterization and bounds of all parameters to be estimated, are presented in Appendix B to ensure reproducibility. We summarize here the most important facts:

Table 2: List of methods/kernels for all experiments.

Category	Name	Description
Hypersphere	Ho	Homoscedastic, only positive correlations
	Ho_NC	Homoscedastic, allowing negative correlations
	He	Heteroscedastic, only positive correlations
	He_NC	Heteroscedastic, allowing negative correlations
	Ho_2	Homoscedastic rank 2, only positive correlations
	Ho_3	Homoscedastic rank 3, only positive correlations
	EHH	Exponential Homoscedastic Hypersphere
LVGP	LVGP	Latent variables (dimension 2)
CS	CS	Compound symmetry
One-hot	one_hot	One-hot encoding
No cat	no cat	Only continuous variables

Table 3: List of kernels for experiments with known group structure only.

Category	Name	Description
Nested Kernels Known groups		Between / Within
	Nested_CS_He	CS / Heteroscedastic
	Nested_He_He	Heteroscedastic / Heteroscedastic
	Nested_Ho_CS	Homoscedastic / CS
	Nested_CS_CS	CS / CS

- For continuous inputs, we use an anisotropic RBF kernel with one lengthscale per dimension
- For categorical inputs, each considered kernel has its own set of parameters
- The set of all parameters involved in the kernels (continuous and categorical) are estimated by maximizing the marginal log likelihood, except for the kernel variance which is obtained in closed-form.
- Optimization is performed with the L-BFGS-B algorithm (Liu and Nocedal, 1989), with scipy’s optimize function (Virtanen et al., 2020). Crucially, and unlike previous published benchmarks, we consider two different optimization scenarios:
 1. A ”short” optimization setting, where we use all default options for scipy’s optimize function, with a maximum of function evaluations set to 15000 and a tolerance equal to $1e^{-9}$. When the number of parameters is large, as in hypersphere kernels, such function evaluation restriction severely impacts the optimizer capacity to reach a local optimum. Indeed, gradients are estimated by finite-differences, thus requiring a potentially large number of function calls at each iteration, and subsequently limiting the allowed number of iterations.
 2. A ”long” optimization setting, where we instead fix a maximum number of iterations equal to 3000, and adapt the maximum number of function calls accordingly, i.e. we set it to $3000 \times (\text{number of hyperparameters} + 1 + \text{number of line search steps})$. We fix the number of line search steps to 20, and lower the tolerance to $1e^{-10}$.

All the results given in Sections 3 and 4 correspond to the ”long” optimization setting, but in some paragraphs we will also present experiments with the ”short” optimization setting, since it will help shed light on interesting facts from a practical perspective.

3.2. Presentation of the datasets

Table 4 presents the datasets used in the study with known groups. For all datasets, designs of experiments of different sizes are generated, and 50 independent replications are created to form the training sets for a given dataset.

Each dataset is presented in detail in Appendix A.

Table 4: Description of datasets with known groups: number of continuous and categorical variables (numbers of levels in parenthesis), and size of training and test datasets. Groups=True means that the groups are known.

Name	Cont	Cat	Train	Test	Groups	Source
f_1	1	1 (13)	39/78/117/156/195	1001	True	Roustant et al. (2020)
f_2	1	1 (10)	30/60/90/120/150	1000	True	Roustant et al. (2020)
Beam bending	2	1 (12)	36/72/108/144/180	1000	True	Roustant et al. (2020)

3.3. Comparison of the methods

The main criterion for assessing the quality of predictions is the Relative Root Mean Squared Error (RRMSE), defined for N^* test ground truth scalar values $(y^{(i)})_{i=1}^{N^*}$ and their associated predictions $(\hat{y}^{(i)})_{i=1}^{N^*}$ as:

$$\text{RRMSE}\left((y^{(i)})_{i=1}^{N^*}, (\hat{y}^{(i)})_{i=1}^{N^*}\right) = \sqrt{\frac{\sum_{i=1}^{N^*} (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{N^*} (y^{(i)} - \bar{y})^2}},$$

where $\bar{y} = \frac{1}{N^*} \sum_{i=1}^{N^*} y^{(i)}$ is the mean of the true values.

In the following, we report two complementary ways to evaluate the methods: individual performances on each dataset, as done usually in such benchmarks, but also performance profiles inspired by the optimization community.

Results on each dataset.

Since there are many dataset variants, we limit the discussion to medium-sized datasets with 6 samples per level (training size equal here to 6 times the number of levels, as there is only one categorical variable in all datasets of Table 4). The results for other dataset sizes are available in the repository containing the code. Figure 2 highlights the results of all compared methods on the f_1 , f_2 and beam bending problems, where the groups are known.

On average, compound symmetry and one-hot encoding have low performance, except for f_2 . Hypersphere kernels have surprisingly good performance, which is due to the long optimization setting. In some instances, low-rank homoscedastic parameterizations perform well, but there is variability depending on the test cases. Finally, consistent good performance is achieved by LVGP and EHH, but Nested_He_He systematically leads to better RRMSE scores.

In order to better analyze all the results across several replications, training set sizes and test cases, we introduce below a new way to compare methods.

Performance profiles.

Performance profiles are inspired by profiles used in optimization benchmarks (Dolan and Moré, 2002), see Appendix C for a detailed introduction. Roughly speaking, the performance profile for a method can be seen as the cumulative distribution function for a performance metric which is the RRMSE in this section. It is a powerful visualization tool to aggregate results obtained on several datasets, with different replications and an ensemble of methods. A performance function $p_i : [0, 1] \rightarrow [0, 1]$ is associated to each method i . Imagine we have access to the scores of every method on all experiments on every dataset. We want to be able to determine if a method is often among the best methods on a given dataset, or if, on the contrary, it is among the worst. We should additionally take into account the scores of all experiments on each dataset. To do so, for each dataset, we first sort the scores of all methods on all experiments. Then, given a performance level $\tau \in [0, 1]$, and a method i , $p_i(\tau)$ counts the proportion of times when an (experiment, method) pair is in the top $\tau\%$ performing methods of all experiments carried out on the given dataset. Such curves are non-decreasing, with $p_i(0) = 0$ and $p_i(1) = 1$. If a method outperforms the other ones on all datasets, then $p_i(\tau)$ is close to 1 even for small values of τ , so the curve grows rapidly and reaches the upper left corner. Conversely, a method that underperforms on all datasets grows slowly at the beginning, approaching the lower right corner. The overall performance of method i can finally be measured by the Area Under The Curve (AUC) of its performance profile plot: $\text{AUC}(p_i) = \int_0^1 p_i(\tau) d\tau$. The performance profile plots gather the curves p_i between 0 and

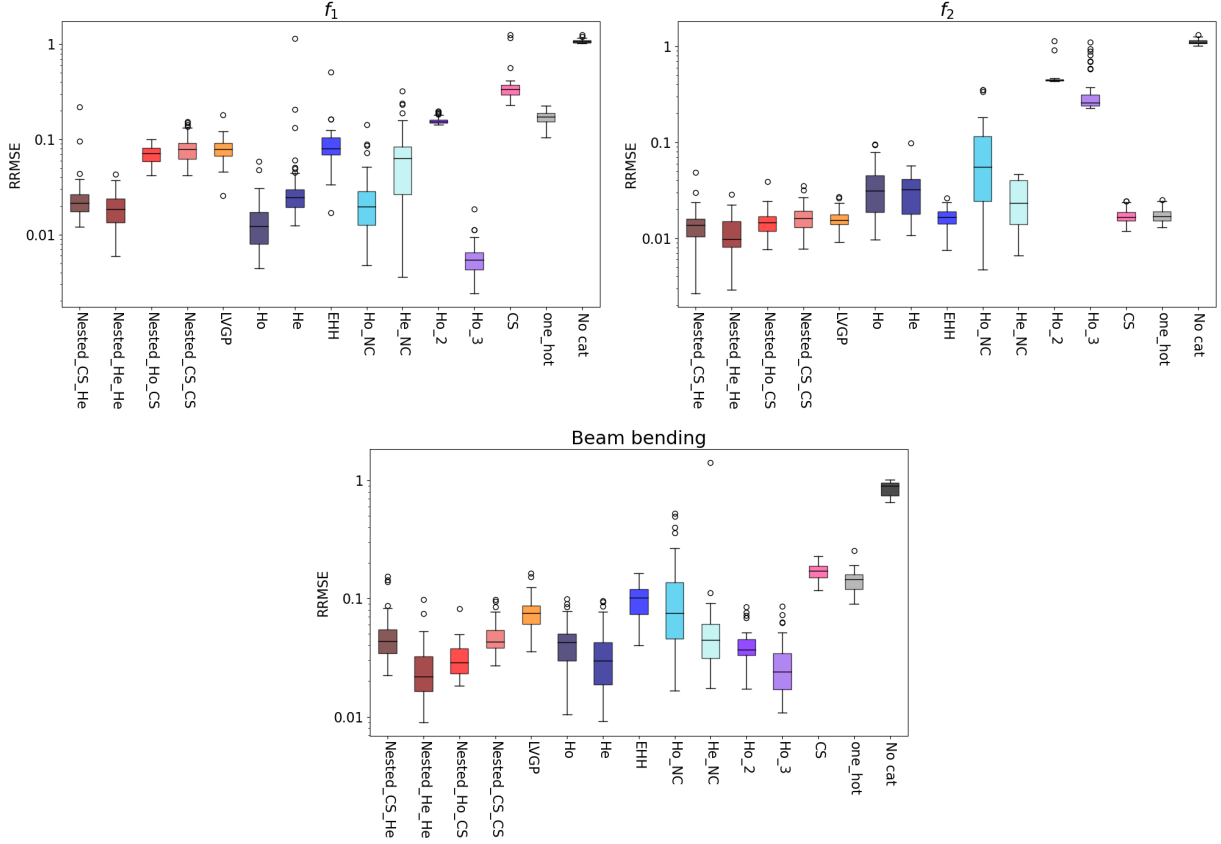


Figure 2: Three datasets with known group structure. Boxplots of the RRMSE over the 50 experiments for all methods, "long" optimization setting.

1 for all methods, and should be read in conjunction with the individual performance figures for each dataset from Section 3.3.

Figure 3 gathers such performance profiles, with method ranked according to their AUC. In the legend, some of them are grouped, which means that their performance are not statistically different from a one-sided Wilcoxon signed rank test. We first observe that group kernels clearly outperform other kernels when there is a known group structure. The 4 types of within/between covariance structures are among the methods with the best overall performance on these datasets. Choosing hyperspheres for the between and within correlations gives better overall results compared to the case where exactly one of them is a CS. Thanks to the long optimization setting, Ho and He reach good performance, low-rank versions of Ho (Ho_2 and Ho_3) give poorer results than classic Ho, which is not surprising. The parameterization specific to EHH also does not seem to improve the performances of the hypersphere. In all cases, LVGP delivers strong results. In Section 4, we provide an analysis for datasets with no known groups which confirms these observations.

4. Experiments with no group structure

In our first experiments with a known group structure, we have clearly illustrated that nested kernels are superior to all other methods by a large margin. In the test cases without such structure, which we would like to investigate now, they are unfortunately inapplicable. We first discuss a new and computationally cheap approach to estimate groups from such datasets, based on target encoding. Importantly, our approach can be applied to cases where it is suspected that there is a group structure but it is unknown, but also to cases where there may be no underlying subdivision.

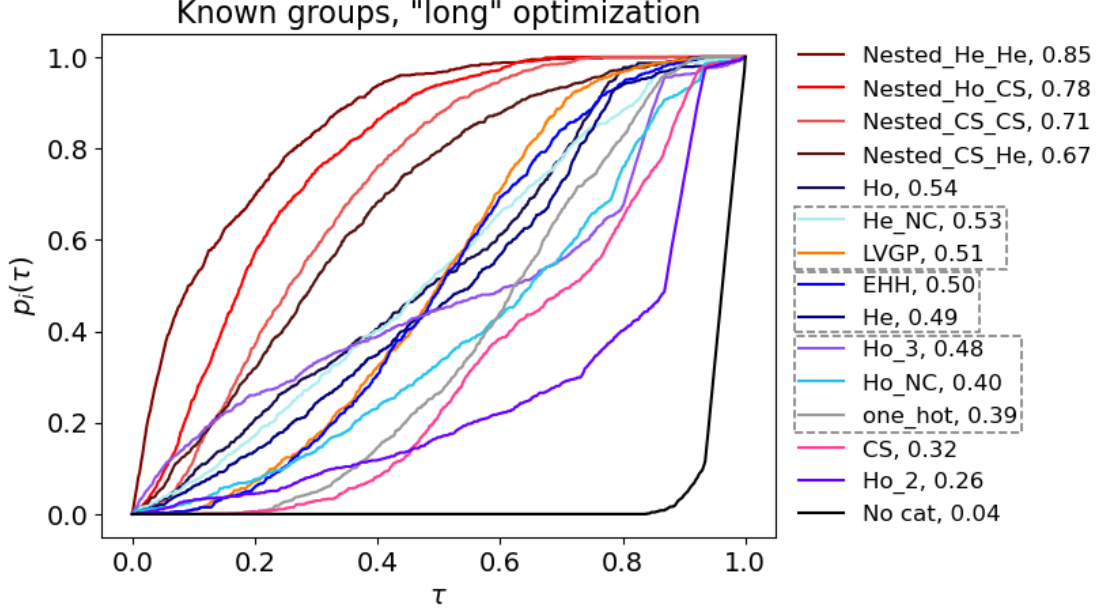


Figure 3: Performance profiles using the RRMSE for datasets with a group structure, "long" optimization setting. The score after the name of the method i is $AUC(p_i)$, methods are sorted in AUC decreasing order.

We then perform the same experimental protocol as in the previous section, but this time on datasets without a group structure, and where we apply the nested kernel methods with the group structure obtained through our estimation procedure.

4.1. Inferring the group structure from target encodings of the levels

In this section, we investigate how to define groups of categorical variables in order to use the nested kernels when there is no prior knowledge about them. We suppose we have access to a single categorical variable (if several categorical variables are present, groups can be selected for each dimension separately). Roustant et al. (2020) suggest to train first a Gaussian process with another categorical kernel $\mathbf{T}^{\text{prox}}$. Then, given a number of clusters between 2 and $C - 1$, a clustering algorithm can be applied with the following distance:

$$d(z, z') = \left(\mathbf{T}_{z,z}^{\text{prox}} + \mathbf{T}_{z',z'}^{\text{prox}} - 2\mathbf{T}_{z,z'}^{\text{prox}} \right)^{1/2}, \quad (8)$$

that is the pseudo-distance induced by the kernel associated to $\mathbf{T}^{\text{prox}}$ (Phillips and Venkatasubramanian, 2011). In practice, they propose to use hierarchical clustering, while the proxy categorical kernel can be chosen as a low-rank kernel such as LVGP. Unfortunately, the impact of this initialization is neither discussed nor illustrated numerically in Roustant et al. (2020). In fact, $\mathbf{T}^{\text{prox}}$ may already be sufficient to reach the desired precision, so there is no need to apply the nested kernel method. On the other hand, if it has a poor predictive performance, we expect a misrepresentation of the distances between levels that do not show clear groups, and the final nested kernel obtained after clustering may be not adapted to the task. A second question concerns the number of groups. It is suggested to choose it as a parameter to be selected from an exhaustive grid search, for example with cross-validation. In addition to requiring several potentially costly training phases, it is common to handle small training sets, which limits cross-validation capabilities. In the following, we present a new approach that consists in representing categorical variables through their target encoding.

Target encoding of categorical variables.

Recall that we consider a training set with categorical inputs $z^{(1)}, \dots, z^{(N)} \in \llbracket C \rrbracket$ and output scalars $y^{(1)}, \dots, y^{(N)} \in \mathbb{R}$.

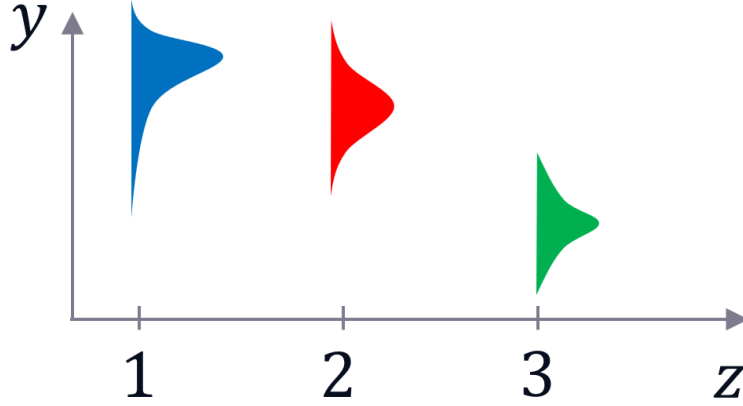


Figure 4: Target encoding illustration. The z -axis represents 3 levels, and for each of them the conditional distribution of the output is represented on the y -axis. Intuitively, levels 1 and 2 should be grouped together since their target encoding distributions are close.

For a level $c = 1 \dots, C$, we consider the distribution of the output y conditioned on the input variable z being equal to c . This distribution can be approximated by the empirical distribution of the observations, i.e. $\nu_c := \frac{1}{| \{i: z^{(i)} = c\} |} \sum_{i: z^{(i)} = c} \delta_{y^{(i)}}$.

Such representation is called a target encoding of the level, since it contains a quantitative feature summary of the level obtained through the output values (the target). We can thus transfer the problem of defining a distance between levels as in (8) to the much more manageable problem of computing a dissimilarity between the encodings. In particular, for empirical distributions, we can use any statistical divergence, including the Wasserstein distance (Peyré et al., 2019) or the Maximum Mean Discrepancy (Gretton et al., 2012), see for example Da Veiga (2025). Intuitively, we measure the proximity of levels by their impact on the output. Figure 4 gives an idealized illustration of target encoding.

In practice however, using classical divergences may not be sufficient when the support of the empirical distributions is small. For most datasets, we work with less than a dozen of samples by level, meaning that estimation of divergences will therefore not be relevant. We instead choose to use simple features of the distributions, namely their mean and standard deviation. The level c is thus embedded in $\mathbb{R} \times \mathbb{R}_+$ as $\psi(c) := (\mu_c, \sigma_c)$ where $\mu_c := \frac{1}{| \{i: z^{(i)} = c\} |} \sum_{i: z^{(i)} = c} y^{(i)}$

and $\sigma_c := \sqrt{\frac{1}{| \{i: z^{(i)} = c\} |} \sum_{i: z^{(i)} = c} (y^{(i)} - \mu_c)^2}$. We call this representation target MSD (Mean and Standard Deviation). Note the similarity with a two-dimensional representation from LVGP, but our proposal is weakly supervised and thus can be computed with almost no computational cost.

Clustering with automatic selection of the number of groups.

Suppose now we have access to a distance matrix $\mathbf{D}^{\text{prox}} \in \mathbb{R}^{C \times C}$. It can either be a distance induced by a categorical kernel $\mathbf{T}^{\text{prox}}$ or a Euclidean distance matrix when levels are embedded in a Euclidean space such as with our proposed target encoding. The first step is to cluster the levels in Q groups from their distances $\mathbf{D}^{\text{prox}}$: here we use agglomerative hierarchical clustering (Murtagh and Contreras, 2012) but any other clustering algorithm may be considered. The remaining question is how to choose the number of groups Q . Instead of training a GP with a group kernel several times with different numbers of clusters $2 \leq Q \leq C - 1$, we suggest instead to use a heuristic. In our experiments, we use the mean Silhouette Coefficient (Rousseeuw, 1987), which measures how well each data point lies within its cluster, and how well separated the clusters are from each other. Other options are discussed in Appendix D, and in particular how to handle $Q = 1$ or $Q = C$ clusters.

Visualization of the group initialization strategies..

For illustration, we compare initialization strategies using latent variables and target encoding. We consider the beam bending problem (see Appendix A) with varying dataset size, and we always take the first experimental seed. We recall that there are 3 groups of levels corresponding to the different filling configurations of the shapes. We start by representing the empirical distributions of the outputs for each level in Figure 5.

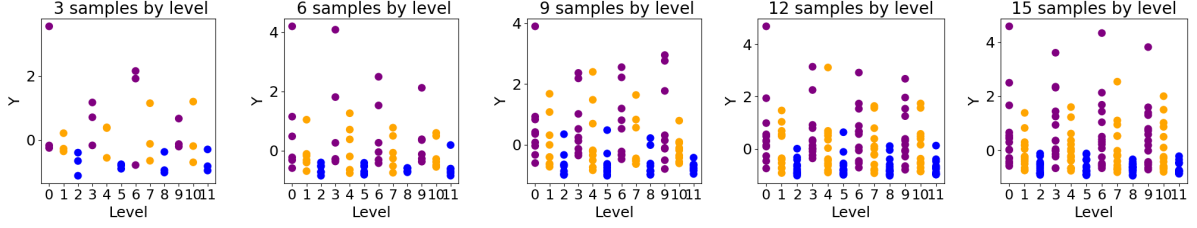


Figure 5: Target distributions of the 12 levels of the beam bending problem. From left to right: 3/6/9/12/15 samples by level. Each color represents a different group.

We then compute the means and standard deviations of the conditional distributions to embed each level in a 2D space. We compare such embeddings with the latent variables obtained after training LVGP in Figure 6, and also show the obtained clusters after running the clustering procedure introduced before. We first note that LVGP fails to identify the true groups regardless of the number of samples per level. For target encoding, the clustering quality increases with the number of samples per level. From 9 samples by level and onward, it can retrieve the true groups successfully. For 3 and 6 samples by level, the group $\{2, 5, 8, 11\}$ is well identified, the group $\{1, 4, 7, 10\}$ is partially identified, and the group $\{0, 6, 3, 9\}$ is harder to detect. Looking back at the representations of the target distributions in Figure 5, this is not surprising: with few samples per level, groups are less obvious to identify.

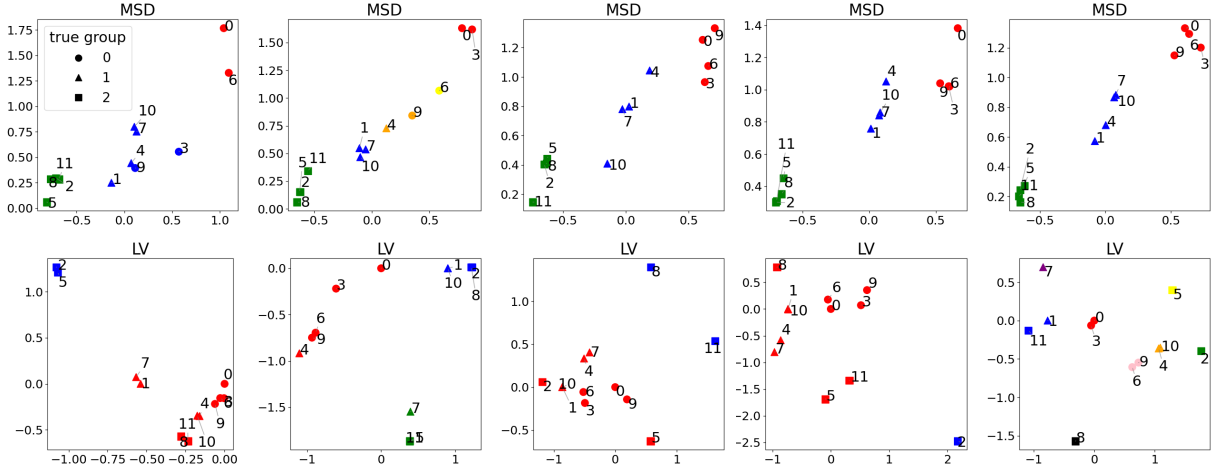


Figure 6: Clustering of the levels based on their target encoding with mean and standard deviation (top) or their LVGP latent variables (bottom) for the beam bending dataset. From left to right: 3/6/9/12/15 samples by level. Colors represent clusters, and the marker styles give the true group.

Finally, if we investigate again our previous numerical experiment on the beam bending dataset, but this time with a nested kernel where the groups are estimated by LVGP or target encoding, we see that a good cluster initialization can help improve scores for this experiment, as illustrated in Table 5. As expected from the clustering quality discussed above, target encoding with MSD is able to reach the same accuracy as if we knew the groups beforehand. Extensive comparisons on other datasets with known groups between the original nested kernels and the ones where the groups are estimated can be found in Appendix D.

In what follows, for datasets with no group structure, we will then use these estimated nested kernels instead of the original ones, see Table 6 for a summary.

Table 5: Comparison of the RRMSE for the Nested_He_He kernel on one experiment of the beam bending problem. Each line corresponds to a group selection strategy.

Samples by level	3	6	9	12	15
True	9.81×10^{-2}	2.04×10^{-2}	8.73×10^{-3}	6.79×10^{-3}	2.04×10^{-3}
MSD	2.00×10^{-1}	1.70×10^{-2}	8.73×10^{-3}	6.79×10^{-3}	2.04×10^{-3}
LV	2.67×10^{-1}	8.16×10^{-2}	1.31×10^{-2}	8.54×10^{-3}	1.7×10^{-2}

Table 6: List of kernels for experiments with no group structure only.

Category	Name	Description
Nested Kernels	Nested_[BETW]_[WITH]_LV	LVGP to initialize groups
Estimated groups	Nested_[BETW]_[WITH]_MSD	Target mean and sd to initialize groups

4.2. Presentation of the datasets

Table 7 presents the datasets with no group structure used in the study. Like in the previous experiments, for all datasets, designs of experiments of different sizes are generated with 50 independent replications. The datasets we select cover a wide range of situations: some have 1 or 2 categorical variables with numbers of levels ranging from 3 to 24, and with different training sizes.

Table 7: Description of all datasets with no group structure: number of continuous and categorical variables (number of levels in parenthesis) and size of training and test datasets. Groups=False means that the groups are unknown.

Name	Cont	Cat	Train	Test	Groups	Source
Borehole	6	2 (3-4)	36/72/108/144/180	1008	False	Zhang et al. (2020)
Borehole2	6	1 (12)	36/72/108/144/180	1008	False	Zhang et al. (2020)
OTL	4	2 (4-6)	72/144/216	1008	False	Zhang et al. (2020)
OTL2	4	1 (24)	72/144/216	1008	False	Zhang et al. (2020)
Piston	5	2 (5-3)	45/90/135	1005	False	Zhang et al. (2020)
Piston2	5	1 (15)	45/90/135	1005	False	Zhang et al. (2020)
Goldstein	2	1 (9)	27/54/81/108/135	999	False	Pelamatti et al. (2021)

4.3. Comparison of the methods

As before, we use the RRMSE as an evaluation metric and compare methods on individual datasets and through performance profiles. In addition, we also propose a new metric that balances accuracy and running time, since this may be of interest in practice to choose a method.

Results on each dataset.

Results provided in the repository containing the code show that on average the nested kernels still perform better than competitors. Here, we will only comment on specific experiments that we think are of practical interest. Looking closely at the dataset list in Table 7, one can observe that we have considered variants of the original OTL, Piston and Borehole dataset, where categorical inputs are either split (original test case) or fused into a single new categorical input with more levels (variant). We proposed to investigate these variants because, in practice, one may wonder if merging all categorical inputs may be beneficial for training a GP. Figure 7 shows that, for the three datasets, it seems preferable to consider the categorical variables separately rather than merging them. The difference is more pronounced for the OTL dataset. Performance is also generally most degraded for hypersphere parameterizations, which suffer from an increase in the number of levels.

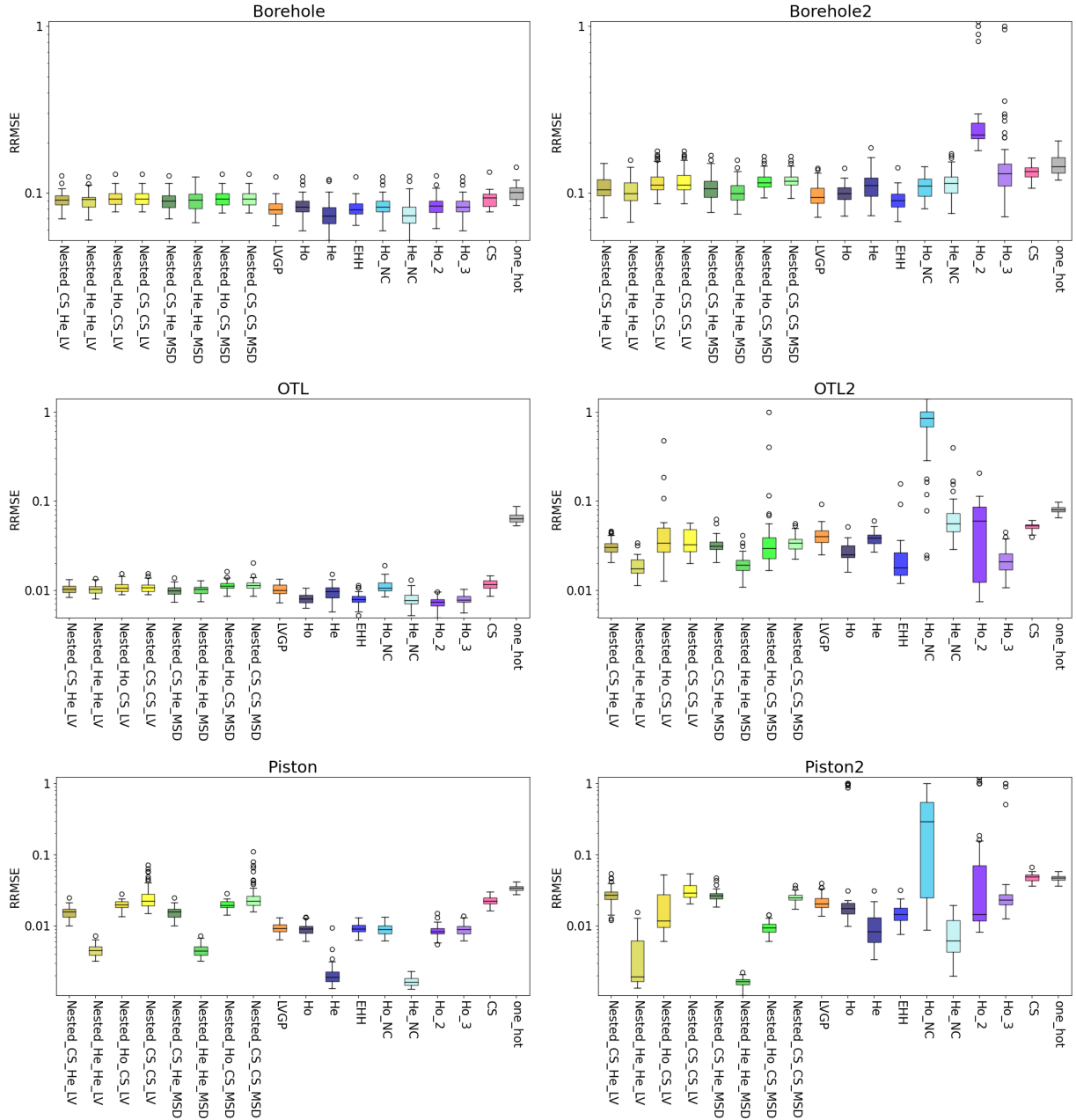


Figure 7: OTL, Piston and Borehole dataset: categorical variables separated (left) and fused (right). Boxplots of the RRSME over the 50 experiments for all methods, "long" optimization setting.

Performance profiles.

We now compute the performance profiles for all methods, they are given in Figure 8a for the long optimization setting and in Figure 8b for the short optimization setting.

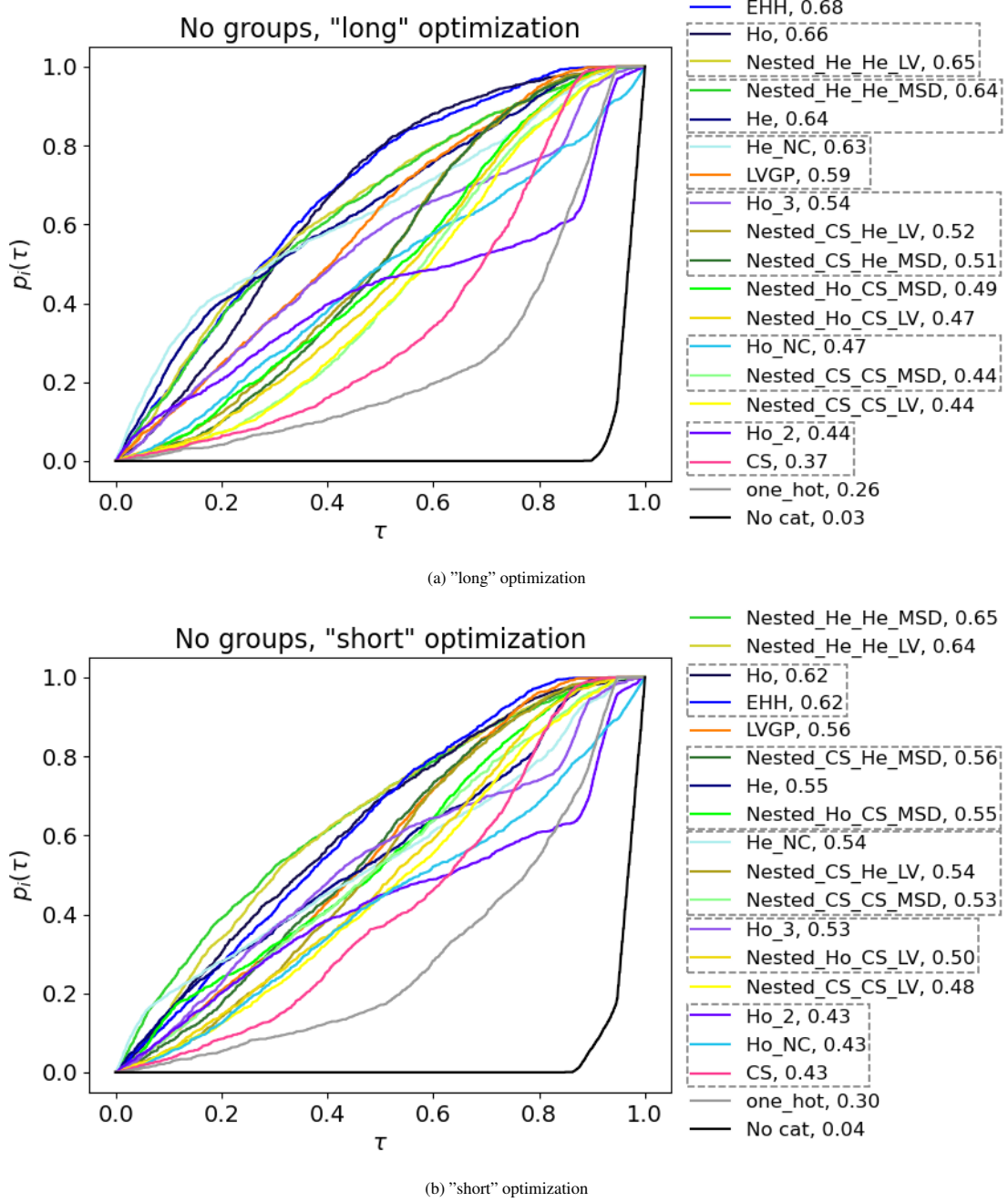


Figure 8: Performance profiles using the RRMSE, "long" (top) and "short" (bottom) optimization setting. Only datasets with no groups are considered. The score after the name of the method i is $AUC(p_i)$, methods are sorted in AUC decreasing order.

In contrast to the results from Section 3, EHH, Ho, He and He_NC yield remarkably strong performance. This finding differs from initial expectations, as hypersphere models are often ignored in comparative studies due to lower-quality results in previous studies. Based solely on these indicators, we should conclude that hyperspheres are preferable to datasets without known groups. However, if we change the optimization options, we arrive at a very different conclusion. In Figure 8b we use the "short" optimization instead of the "long" optimization setting. Remember that the "short" optimization relies on the default options from scipy's L-BFGS-B algorithm as we detailed in Section

3.1. With these parameters, Nested_He_He with automatic selections via LVGP or target encodings outperforms other models on datasets with unknown groups. As discussed in more detail in Appendix E, this difference in rankings between "long" and "short" optimizations can be explained by a significant deterioration in the scores of hypersphere models. It is therefore possible to obtain very good scores for hyperspheres for these datasets, but this takes hours instead of a few minutes of training, which is unfortunately not feasible for standard use cases. When using default optimization options and thus manageable run times, we realize that nested kernels offer the most robust alternative.

We find that group initializations with LVGP or MSD are nearly equivalent. However, unlike in the case of known groups, the choice of between and within covariance structures is more pronounced. For hypersphere models, Ho_NC and low rank variants are once again inferior to other full-rank variants. Nevertheless, EHH parameterization seems to be more useful on these datasets. Lastly, while LVGP is still strong, it appears clearly that strategies based on one-hot encoding or CS should be avoided because they show degraded performances.

Finally, we also provide in Figure 14 from Appendix C the performance profiles when we include all datasets (with or without groups). It can be seen that Nested_He_He_LV and Nested_He_He_MSD achieve the best overall performance, even in the case of "long" optimization.

Tradeoff between computation time and RRMSE.

In previous representations, only the RRMSE was taken into account. However, computation time is also an indicator that can influence the choice of the kernel in practical studies, as was already touched upon with both optimization settings. We now also investigate time-based indicators by evaluating the sequential time required to train the GP model. As in the case of RRMSE, performance profiles can be defined for methods based on running times across all experiments. Computation times for nested methods with automatic group initializations (LV and MSD) do not include the identification of groups (this step is very fast thanks to the target encoding of the MSD representation contrary to LV). Importantly, for this paragraph and study only, we consider all possible datasets, and the methods available on each one of them. In Figure 9, each method is represented by a point, with the x-axis corresponding to the AUC of its performance profile based on RRMSE, and the y-axis to the AUC of its time-based performance profile. A good method is therefore located as far up on the right as possible. Observe first that rankings based on computation times strongly correlate with the numbers of parameters depicted in Figure 1.

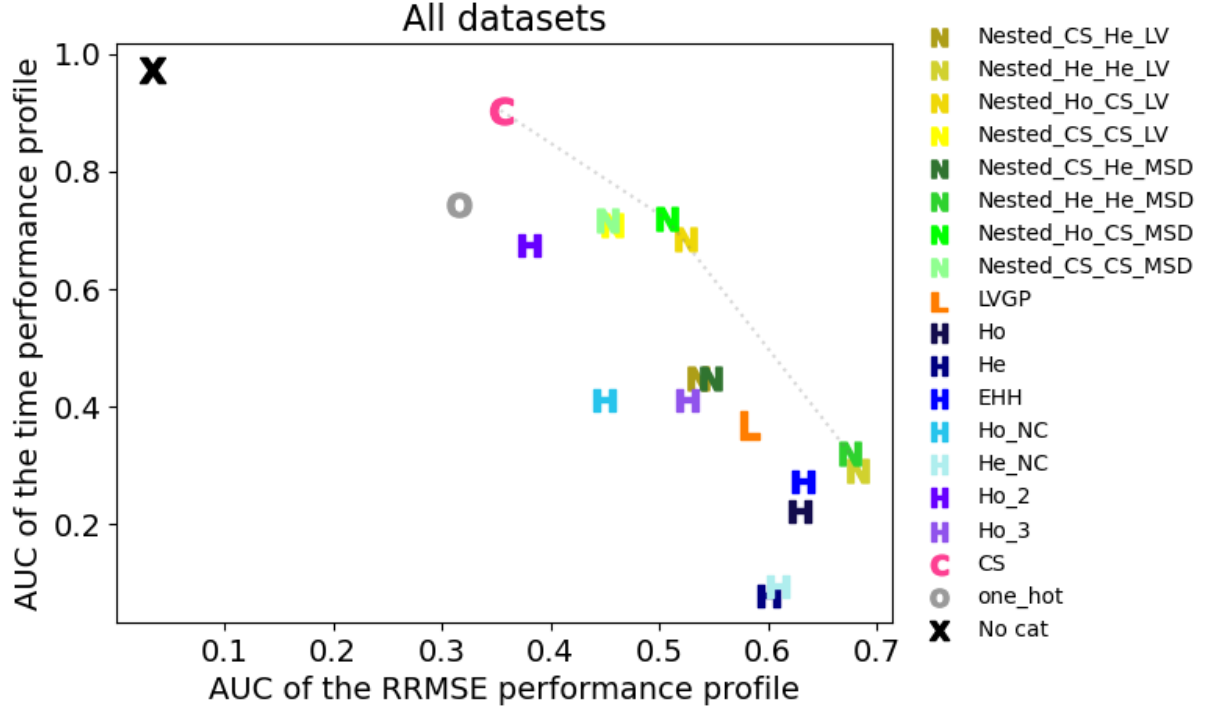


Figure 9: AUC based on sequential time versus AUC based on RRMSE, "long" optimization setting. The dotted grey line represents the "Pareto front".

One-hot encoding and compound symmetry have low computation times, but they have the worst prediction capabilities. On the contrary, hypersphere models have the highest computation time, and average prediction capabilities. Using low-rank hypersphere variants naturally lowers the computation time, but it decreases the quality of predictions. Nested kernels having both a hypersphere as within and between correlation have times similar to hyperspheres, but they offer better predictive performance. The computation times of LVGP are rather high, but it is still a reliable approach that is independent on prior group structure. We also represent in Figure 9 a "Pareto front", which indicates methods that achieve the best trade-offs between time and learning capacity. Note that this is mainly for illustration purposes, as the lines have no formal definition. This suggests to use nested kernels with automatic groups and heteroscedastic models as between and within covariances for the best learning capacity, but at the cost of a higher computation time. One may replace the between covariances with a homoscedastic structure to reduce the complexity, while keeping average performances. Finally, the CS model seems to offer the best performance when time is even more limited.

5. Conclusion

In this paper, we detail an extensive comparative study of categorical kernels used for Gaussian process regression in mixed spaced (with continuous and categorical input variables). 23 categorical kernels are evaluated on 42 datasets widely used in the literature, with two optimization settings. To our knowledge, this is the first time such an advanced comparison of the different kernels found in the literature is carried out, with a ranking of methods based on new performance metrics on several dataset experiments. In addition, our study is entirely reproducible with the accompanying code.

We also propose an innovative way of defining groups of levels when they are not known or when there is no prior information on the group structure, using clustering based on a target encoding representation of the levels. This new

representation delivers performance comparable to that based on LVGP initialization, but without the need to pre-train another model. It is clear from our study that nested kernels strategies outperform other kernels in terms of model error when groups exist and are known. Even in the absence of known groups, nested variants with automatic group selection using LVGP or target encoding are once again among the best methods. However, the choice of within and between correlation types in nested kernels matters, He/He being the best choice but at the cost of larger computing times. Contrary to the popular belief, hypersphere models are competitive from the perspective of performances, but their high computation time is not cost-effective when compared to other methods. They present good performances only when the hyperparameters of the optimization routine (tolerance, maximal number of evaluations) are chosen beyond their default values. Low-rank variants of hypersphere do not appear to be a reliable alternative, and also suffer from instabilities, in particular with negative correlations, but other parameterization choices like EHH can somewhat improve the performances. CS and one-hot encoding should not be used if good performance is required, while LVGP is always among the best methods.

Acknowledgements

This work was partially supported by the Agence Nationale de la Recherche through the SAMOURAI (Simulation Analytics and Metamodel-based solutions for Optimization, Uncertainty and Reliability Analysis) project under grant ANR20-CE46-0013 and the EXAMA (Methods and Algorithms at Exascale) project under grant ANR-22-EXNU-0002.

Appendix A. Presentation of the datasets

All training and test datasets except Goldstein are generated in the same way: for categorical variables, the same number of samples is used for each level (or tuple of levels in the multivariate case), and for continuous variables we create a Sliced Latin Hypercube Design (SLHD) (Qian, 2012). 50 independent replications are created to form the training sets for a given dataset. Remark that the designs are not optimized using e.g. a maximin LHS (Stein, 1987) since we want to keep some variability between experiments. For Goldstein, since the continuous variables are subject to constraints, we generate points by a rejection procedure combined with a uniform distribution. In practice, a standard scaling is applied to all outputs as well as all input continuous variables. For the datasets having two categorical variables (OTL, Piston and Beam Bending), we also consider variants of the dataset where these levels are merged to form a single categorical variable.

Analytical function f_1 .

The function is defined for a continuous variable and a categorical input with 13 levels as follows:

$$f_1(x, z) = \cos(7\pi \frac{x}{2} + (0.4 + \frac{z}{15})\mathbb{1}_{\{z > 9\}} - \frac{z}{20})$$

where $x \in [0, 1]$, $z \in \llbracket 13 \rrbracket$. There are two groups of levels ($\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $\{10, 11, 12, 13\}$) that appear clearly in Figure 10. We consider 5 different sizes for the training sets 39/78/117/156/195 corresponding to 3/6/9/12/15 samples by level, respectively.

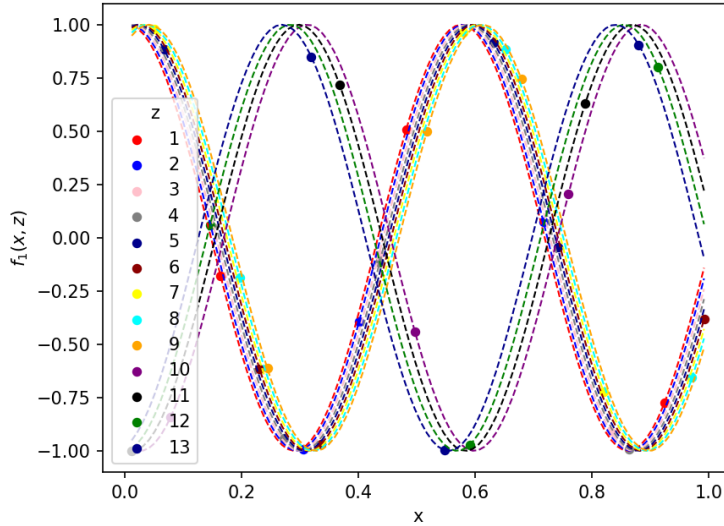


Figure 10: f_1 . Bullet points represent training samples, and lines represent the true functions for each level.

Analytical function f_2 .

The function is defined for a continuous variable and a categorical input with 10 levels as follows:

$$f_2(x, z) = \begin{cases} x + 0.01(x - \frac{1}{2})^2 \frac{z}{10} & \text{if } z = 1, 2, 3, 4 \\ 0.9 \cos(2\pi(x + (z - 4)\frac{z}{20}))e^{-x} & \text{if } z = 5, 6, 7 \\ -0.7 \cos(2\pi(x + (z - 7)\frac{z}{20}))e^{-x} & \text{if } z = 8, 9, 10 \end{cases}$$

where $x \in [0, 1]$, $z \in \llbracket 10 \rrbracket$. There are three explicit groups of levels ($\{1, 2, 3, 4\}$, $\{5, 6, 7\}$ and $\{8, 9, 10\}$) that appear clearly in Figure 11. We consider 5 different sizes for the training sets 30/60/90/120/150 corresponding to 3/6/9/12/15 samples by level, respectively.

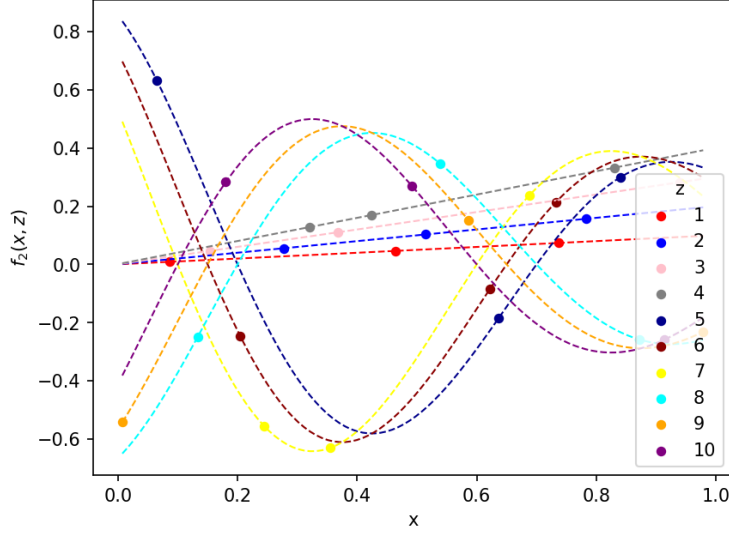


Figure 11: f_2 . Bullet points represent training samples, and lines represent the true functions for each level.

Borehole.

The continuous Borehole function is:

$$f_B(r, T_u, H_u, T_l, L, K_w) = 2\pi T_u (H_u - H_l) \left\{ \ln\left(\frac{r}{r_w}\right) \left(1 + 2 \frac{LT_u}{\ln\left(\frac{r}{r_w}\right) r_w^2 K_w} + \frac{T_u}{T_l} \right) \right\}^{-1}$$

where $r \in [100, 50000]$, $r_w \in [0.05, 0.15]$, $T_u \in [63070, 115000]$, $H_u \in [990, 1110]$, $T_l \in [63.1, 116]$, $H_l \in [700, 820]$, $L \in [1120, 1680]$, $K_w \in [9855, 12045]$, see Morris et al. (1993) for a more precise description of the variables. r_w and H_l are treated as categorical variables with respectively 3 and 4 levels whose associated continuous values are evenly distributed across their definition spaces. This gives a total of 6 continuous and 2 categorical variables. The Borehole2 function in Table 7 corresponds to the same dataset, but the two categorical variables are fused into a single one with 12 levels. We consider 5 different sizes for the training sets 36/72/108/144/180 corresponding to 3/6/9/12/15 samples by level tuple, respectively.

OTL.

The continuous OTL function is:

$$f_O(R_{b1}, R_{b2}, R_f, R_{c1}, R_{c2}, \beta) = \frac{(V_{b1} + 0.74)\beta(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\beta(R_{c2} + 9)}{R_{c1}(\beta(R_{c2} + 9) + R_f)}$$

where $V_{b1} = \frac{12R_{b2}}{R_{b1} + R_{b2}}$, $R_{b1} \in [50, 150]$, $R_{b2} \in [25, 70]$, $R_f \in [0.5, 3]$, $R_{c1} \in [1.2, 2.5]$, $R_{c2} \in [0.25, 1.2]$, $\beta \in [50, 300]$, see Ben-Ari and Steinberg (2007) for a more precise description of the variables. R_f and β are treated as categorical variables with 4 and 6 levels whose associated continuous values are respectively (0.5, 1.2, 2.1, 2.9) and (50, 100, 150, 200, 250, 300). This gives a total of 4 continuous and 2 categorical variables. The OTL2 function in Table 7 corresponds to the same dataset, but the two categorical variables are fused into a single one with 24 levels. We consider 3 different sizes for the training sets 72/144/216 corresponding to 3/6/9 samples by level tuple, respectively.

Piston.

The continuous Piston function is:

$$f_P(M, S, V_0, k, P_0, T_a, T_0) = 2\pi \sqrt{M^{-1} \left(k + S^2 \frac{P_0 V_0 T_a}{T_0} \left\{ \frac{S}{2k} \left(\sqrt{A^2 + 4k \frac{P_0 V_0 T_a}{T_0}} - A \right) \right\}^{-2} \right)}$$

where $A = P_0 S + 19.62 M - \frac{k V_0}{S}$, $M \in [30, 60]$, $S \in [0.005, 0.02]$, $V_0 \in [0.002, 0.01]$, $k \in [1000, 5000]$, $P_0 \in [90000, 110000]$, $T_a \in [290, 296]$, $T_0 \in [340, 360]$, see Sack et al. (1989) for a more precise description of the variables. P_0 and k are treated as categorical variables with respectively 3 and 5 levels whose associated continuous values are evenly distributed across their definition spaces. This gives a total of 5 continuous and 2 categorical variables. The Piston2 function in Table 7 corresponds to the same dataset, but the two categorical variables are fused into a single one with 15 levels. We consider 3 different sizes for the training sets 45/90/135 corresponding to 3/6/9 samples by level tuple, respectively.

Beam bending.

We consider the Euler-Bernoulli beam bending problem (Roustant et al., 2020). Remark that this dataset is different

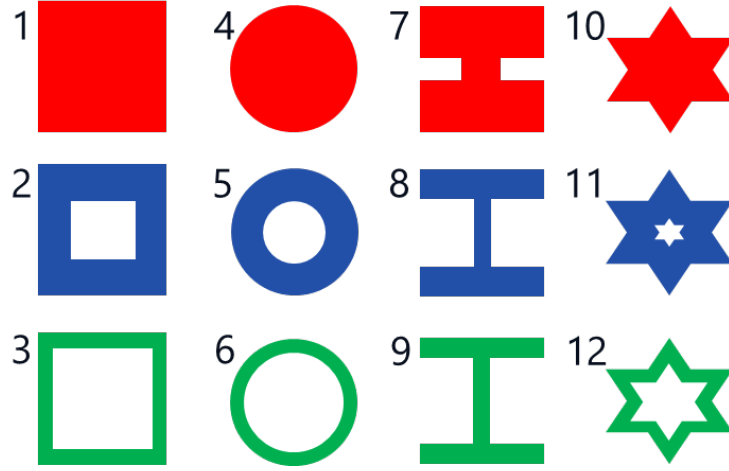


Figure 12: The twelve shapes of the beam bending problem. The three groups can be identified thanks to the colors. Figure inspired by Roustant et al. (2020).

from the one in Zhang et al. (2020). In this problem, there are various shapes with several filling configurations that can be considered as 12 different levels (see Figure 12). 3 groups, each one corresponding to a given filling configuration. The underlying physical problem is the following: fix the beam at one end and apply force at the other. Under linear elasticity assumptions, the beam length at the cross section can be approximated by the following quantity (Timoshenko and Gere, 1997):

$$f_{BB}(L, S, I) = \frac{L^3}{3S^2 I}$$

where $L \in [10, 20]$ and $S \in [1, 2]$, see Roustant et al. (2020) for a more precise description of the variables. I is treated as a categorical variable with the associated continuous values (0.0833, 0.139, 0.380, 0.0796, 0.133, 0.363, 0.0859, 0.136, 0.360, 0.0922, 0.138, 0.369) for all 12 shapes in Figure 12. This gives a total of 2 continuous and 1 categorical variables. We consider 5 different sizes for the training sets 36/72/108/144/180 corresponding to 3/6/9/12/15 samples by level, respectively.

Goldstein.

The Goldstein function (Pelamatti, 2020; Picheny et al., 2013) is defined for two continuous variables and a categorical

variable with 9 levels as follows:

$$\begin{aligned}
f(x_1, x_2, z) = f(x_1, x_2, x_3, x_4) = & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^4 \cdot 10^{-8} \\
& - 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^3 \cdot 10^{-6} \\
& + 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 \\
& - 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 \\
& + 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} \\
& - 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} \\
& - 5.62199x_2x_3x_4 \cdot 10^{-5}
\end{aligned}$$

with $x_3 = 20, x_4 = 20$ if $z = 0$, $x_3 = 20, x_4 = 50$ if $z = 1$, $x_3 = 20, x_4 = 80$ if $z = 2$, $x_3 = 50, x_4 = 20$ if $z = 3$, $x_3 = 50, x_4 = 50$ if $z = 4$, $x_3 = 50, x_4 = 80$ if $z = 5$, $x_3 = 80, x_4 = 20$ if $z = 6$, $x_3 = 80, x_4 = 50$ if $z = 7$, $x_3 = 80, x_4 = 80$ if $z = 8$. where $x_1, x_2 \in [0, 1]$, $z \in \llbracket 9 \rrbracket$. The constraint $g(x_1, x_2, z) = c_1 \sin(\frac{x_1}{10})^3 + c_2 \cos(\frac{x_2}{10})^2 \leq 0$ is used to create the design of experiments, with $c_1 = 2, c_2 = 0.5$ if $z = 0$, $c_1 = 2, c_2 = -1$ if $z = 1$, $c_1 = 2, c_2 = -2$ if $z = 2$, $c_1 = -2, c_2 = 0.5$ if $z = 3$, $c_1 = -2, c_2 = -1$ if $z = 4$, $c_1 = -2, c_2 = -2$ if $z = 5$, $c_1 = 1, c_2 = 0.5$ if $z = 6$, $c_1 = 1, c_2 = -1$ if $z = 7$, $c_1 = 1, c_2 = -2$ if $z = 8$. We consider 5 different sizes for the training sets /54/81/108/135 corresponding to 3/6/9/12/15 samples by level, respectively. Figure 13 gives an illustration of the Goldstein function.

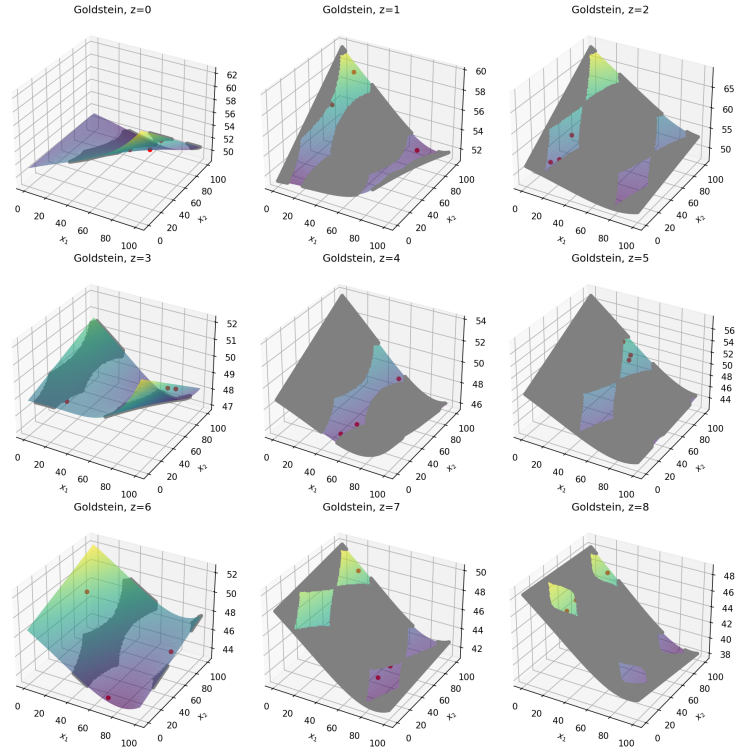


Figure 13: Goldstein. Bullet points represent training samples. Contour plots give the value of the function for each level.

Appendix B. Experimental setup

In this section, details of the experiments are presented to ensure reproducibility. As a matter of fact, we found that several papers commented only on the parameters specific to their methods, and omitted critical details about the optimization.

Continuous kernel.

We use an ARD RBF kernel for the continuous inputs. The RBF kernel is parameterized by the lengthscales. For each coordinate, the upper bound on the lengthscales is selected as two times the maximum distance between two input points in the training set and the lower bound is half the minimum distance between input points in the training set (coordinate-wise). We observed that fixing the minimal lengthscale to an arbitrary small value leads to failures in the optimization. In our code, we also propose another parameterization with the log of precisions giving similar results when setting the limits on the parameters correctly.

Categorical kernels.

For the categorical part, the bounds on the parameters depend on the method. For the hypersphere variants, it is often difficult to know which are the bounds. In fact, we can consider hyperspheres allowing or not negative correlations. For the Hypersphere variants, we thus take an upper bound equal to $\frac{\pi}{2}$ for all methods except Ho_NC and He_NC where we allow negative correlations by taking an upper bound of π . When the kernel is a nested variant, we also use an upper bound of π when between or within kernels are hyperspheres. For LVGP, latent representations have values in $[-3, 3]$. Other categorical kernels do not require to fix their bounds.

Optimization.

Parameters are selected via a maximization of the marginal log likelihood, where we use the closed-form formulas for the mean and global variance parameters of the GP (CF Rasmussen (2003)). Both the nugget and the parameters of the continuous and categorical kernels are optimized with the L-BFGS-B algorithm. The bounds on the nugget are $[1e - 8, 1e - 4]$. We use 96 restart points obtained via a maximin LHS.

Scaling of the data.

In practice, we considered versions of the datasets where both the continuous inputs and the outputs have been scaled to have a mean equal to zero and a variance equal to one.

Appendix C. Performance profiles

1. Presentation of the performance profiles

Let us denote by I the number of methods, J the number of datasets, and K the number of experiments. Let us consider the methods $i = 1, \dots, I$, and the datasets $j = 1, \dots, J$. We have access to several experiments $k = 1, \dots, K$. One specific experiment is associated to a specific design of experiments (and corresponds to one random seed). We observe the score $s_{i,j,k} \in \mathbb{R}$ for the method i on the k -th experiment on the dataset j . Without loss of generality, we assume that the score must be minimized (for instance if the score is a RRMSE). Instead of looking at aggregated scores for each pair (dataset, method) according to all experiments, typically an average or median, we propose to consider each experiment separately.

In our experiments, all the (dataset, method) pairs share the same number of experiments. However, when some method is not applied to a given dataset, we arbitrarily set the score (for all experiments) to ∞ . Given a method i , we can then denote by $\mathcal{J}_i = \{(j, k) \in [J] \times [K] : s_{i,j,k} < \infty\}$ the set of (dataset, experiment) pairs where the method i is evaluated. Similarly, for a dataset j , we can denote by $\mathcal{I}_j = \{(i, k) \in [I] \times [K] : s_{i,j,k} < \infty\}$ the set of (method, experiment) pairs where the dataset j is evaluated.

For a given dataset $j = 1, \dots, J$, we first sort all the scores obtained with all possible experiments and methods $\{s_{i,j,k} : (i, k) \in \mathcal{I}_j\}$. Given a quantile level $\tau \in [0, 1]$, we then define the quantile of order τ of the latter set as

$q_{j,\tau} := \text{Quantile}_\tau(\{s_{i,j,k} : (i, k) \in \mathcal{I}_j\}) := \inf \left\{ t \in \mathbb{R} : \frac{1}{|\mathcal{I}_j|} \sum_{(i,k) \in \mathcal{I}_j} \mathbb{1}_{\{s_{i,j,k} \leq t\}} \geq \tau \right\}$. This quantile in fact corresponds to the $\lceil \tau |\mathcal{I}_j| \rceil$ -th smallest score value for the given dataset over all possible (experiment, method) pairs.

For a given method $i = 1, \dots, I$, we can now measure its global performance thanks to the rankings of the experiments realized with this method on all the datasets. Choosing a performance level $\tau \in [0, 1]$, we introduce the following quantity: $p_i(\tau) = \frac{1}{|\mathcal{J}_i|} |\{(j, k) \in \mathcal{J}_i : s_{i,j,k} \leq q_{j,\tau}\}|$. The latter quantity evaluates the proportion of (experiment, dataset) pairs where the (experiment, method) pair is in the top τ performing methods of all experiments performed on the given dataset. The performance profile plot of a method is then characterized by the function $p_i : [0, 1] \rightarrow [0, 1]$.

2. Additional performance profiles

In Figure 14, we consider all datasets. In comparison to Figure 8a using the same optimization options, the LVGP and MSD automatic variants of nested kernels with He/He achieve even better results and outperform hypersphere models. This additional result is not surprising, as datasets with a group structure are added to this performance profile.

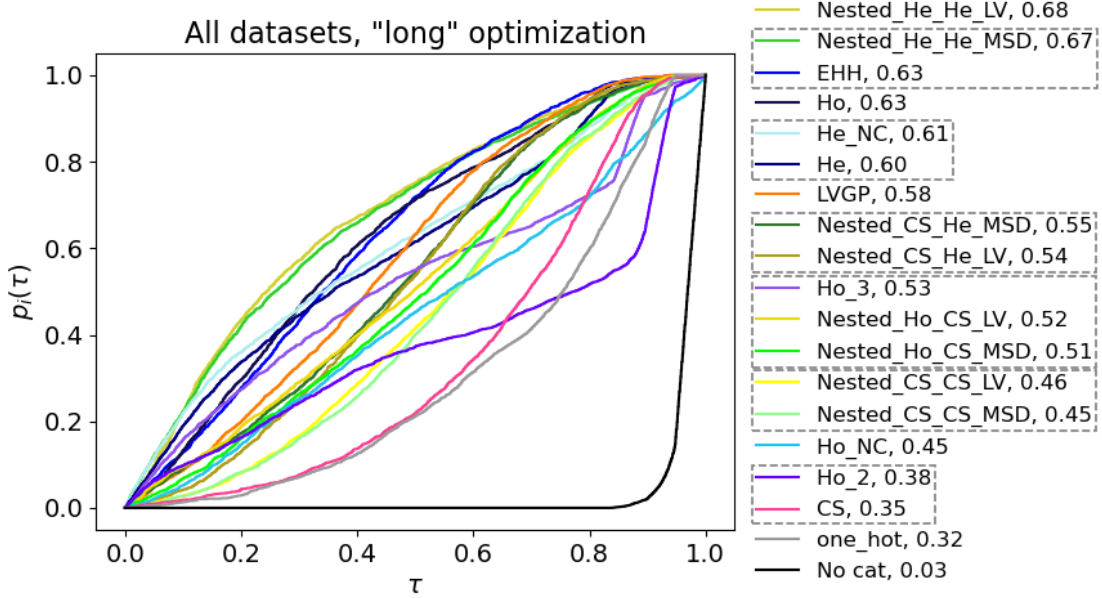


Figure 14: Performance profiles using the RRMSE. All datasets and all kernels are considered. The score after the name of the method i is $\text{AUC}(p_i)$, methods are sorted in AUC decreasing order.

Appendix D. Clustering

In this section, we give more details about the clustering procedure and provide additional results about the impact of group selection on the regression scores.

1. Choice of the number of clusters

A fundamental question when performing clustering is the selection of the number of clusters. Roustant et al. (2020) suggest setting the number of clusters by an exhaustive search. For all sizes $2 \leq Q \leq C - 1$, one can rely on cross-validation to compute a prediction criterion in order to define the best value of Q . This strategy can however be costly because it requires the training of several Gaussian processes. Instead, we propose to launch the categorical GP only once with a pre-determined number of clusters. The selection of the 'optimal' number of clusters remains a topic of ongoing debate, with no universally accepted criterion. Specific methods like Gaussian mixture models employ information criteria to penalize the complexity of the model, while methods such as the elbow plots are more general but harder to apply. Other methods such as the Density-Based Clustering Validation (DBCV) score (Moulavi et al., 2014) are also not suitable, as they require a large number of points to make sense. We recommend using the Silhouette score (Rousseeuw, 1987) which only requires pseudo-distances, unlike some criteria that manipulate data in Euclidean spaces. In the following, we describe the Silhouette score.

Let $2 \leq Q \leq C - 1$ be the number of clusters, and let $C_1, \dots, C_Q$ be a partition of $\llbracket C \rrbracket$ into Q clusters obtained by any clustering algorithm. The cluster of z is denoted as $q(z) \in \llbracket Q \rrbracket$. We suppose we have access to a pseudo-distance

$d : \llbracket C \rrbracket \times \llbracket C \rrbracket \rightarrow [0, +\infty)$. The Silhouette coefficient of $z \in \llbracket C \rrbracket$ is defined as:

$$s(z) = \frac{b(z) - a(z)}{\max(a(z), b(z))}$$

where $a(z) = \frac{1}{|C_{q(z)}|-1} \sum_{z' \in C_{q(z)} \setminus \{z\}} d(z, z')$ is the average distance of the point to its group (with $a(z) = 0$ if $|C_{q(z)}| = 1$), and $b(z) = \min_{i \neq q(z)} \frac{1}{|C_i|} \sum_{z' \in C_i} d(z, z')$ is the average distance of the point from its neighbouring group. The global Silhouette score of the clustering is defined as follows:

$$s_{\text{Silhouette}}(\{C_1, \dots, C_Q\}) = \frac{1}{Q} \sum_{i=1}^Q \frac{1}{|C_i|} \sum_{z \in C_i} s(z)$$

The Silhouette score varies between -1 (worst classification) and 1 (best classification). As we can see from the definition of the Silhouette score, it is naturally only suitable for partitions having between 2 and $C - 1$ clusters. In the case where we want to consider 1 or C clusters, the model in fact corresponds to the CS or hypersphere models. In our experiments, we limit the choice between 2 and $C - 1$ for an automatic selection maximizing the Silhouette score. For a more general case, we encourage preliminary visualizations to identify the presence obvious groups, and, if not, run the Gaussian process regression with a hypersphere model.

2. Regression scores depending on the group selection strategy.

In this section, we consider the RRMSE of the Gaussian process regression using the Nested_He_He graph kernels. We compare three group selection strategies on datasets with known group structures, including the true groups, and groups obtained thanks to clustering using respectively the MSD and LVGP representations. We give the boxplots of the RRMSE for the 50 experiments on the datasets f_1 , f_2 and beam bending with various sizes in Figure 15. First of all, the overall error decreases with the number of samples per level in all cases, which is not surprising. In any case, the nested kernel version with real groups performs best. The automatic selection with LVGP or MSD obtain errors of the same order of magnitude as in the case where true groups are used for all datasets. Selection with MSD seems preferable or equivalent to that of LVGP when the number of samples per level is greater than 6 for f_2 and beam bending, and 9 for f_1 . This supports the observation of Section 4.1 that a sufficient number of samples per level is necessary to take full advantage of representation with target encodings.

Appendix E. Influence of the optimization

As we discussed in Section 4, optimization requires special attention and has a significant impact on the global performance of the models used. This impact is even more visible on hypersphere models, as can be seen in Figure 8. We add the comparison of boxplots the dataset f_1 with 6 samples by level in Figure 16 for the "long" and "short" optimization. RRMSE are severely degraded for all the hypersphere models. Using the homoscedastic hypersphere kernel, performing the 96 restarts of the optimization takes a total time of 842 seconds for the "short" optimization while it takes 9620 seconds for the "long" optimization.

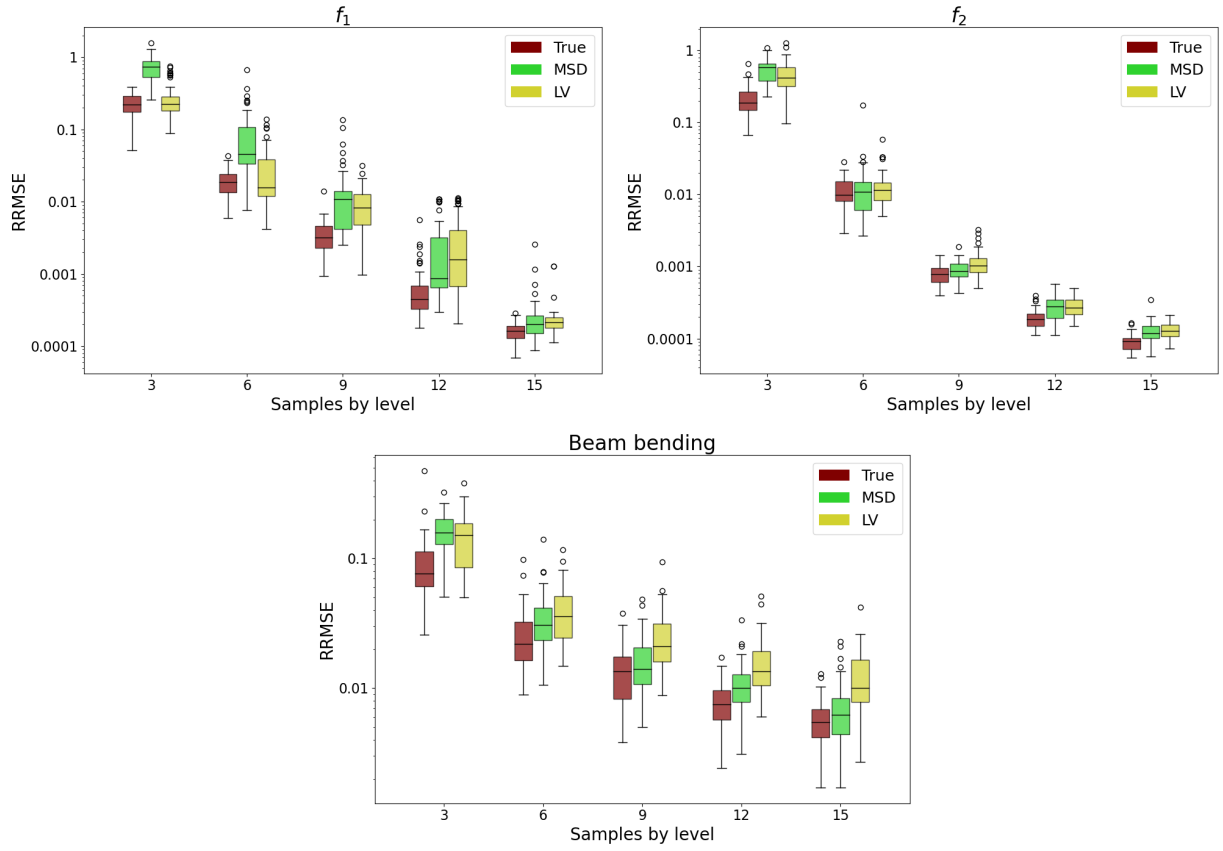


Figure 15: Boxplots of the RRSME over the 50 experiments obtained with the Nested_He_He kernel relying on different group selection strategies. We consider different numbers of samples per level for the three datasets with known group structure.

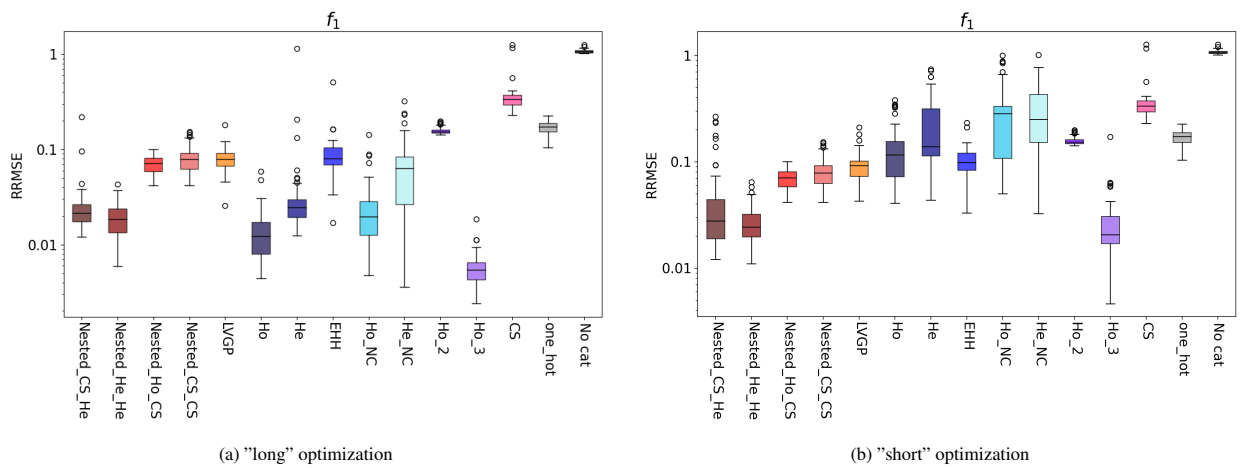


Figure 16: Boxplots of the RRSME over the 50 experiments for all methods on the dataset f_1 with 6 samples by level depending on the optimization options.

References

- Aitchison, J., Aitken, C.G., 1976. Multivariate binary discrimination by the kernel method. *Biometrika* 63, 413–420.
- Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E., 2020. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems* 33, 21524–21538.
- Bartoli, N., Lefebvre, T., Dubreuil, S., Olivanti, R., Priem, R., Bons, N., Martins, J.R., Morlier, J., 2019. Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and technology* 90, 85–102.
- Baudin, M., Dutfoy, A., Iooss, B., Popelin, A.L., 2015. Open turns: An industrial software for uncertainty quantification in simulation. *arXiv preprint arXiv:1501.05242*.
- Beauthier, C., Mahajan, A., Sainvitu, C., Hendrick, P., Sharifzadeh, S., Verstraete, D., 2014. Hypersonic cryogenic tank design using mixed-variable surrogate-based optimization, in: *Engineering Optimization IV—Proceedings of the 4th International Conference on Engineering Optimization, ENGOPT*, pp. 543–549.
- Ben-Ari, E.N., Steinberg, D.M., 2007. Modeling data from computer experiments: an empirical comparison of kriging with mars and projection pursuit regression. *Quality Engineering* 19, 327–338.
- Carpintero Perez, R., Da Veiga, S., Garnier, J., Staber, B., 2024. Gaussian process regression with sliced wasserstein weisfeiler-lehman graph kernels, in: *International Conference on Artificial Intelligence and Statistics*, PMLR. pp. 1297–1305.
- Cuesta Ramirez, J., Le Riche, R., Roustant, O., Perrin, G., Durantin, C., Glière, A., 2022. A comparison of mixed-variables bayesian optimization approaches. *Advanced Modeling and Simulation in Engineering Sciences* 9, 6.
- Da Veiga, S., 2025. Distributional encoding for Gaussian process regression with qualitative inputs. *arXiv preprint arXiv:2506.04813*.
- Da Veiga, S., Gamboa, F., Iooss, B., Prieur, C., 2021. Basics and trends in sensitivity analysis: Theory and practice in R. *SIAM*.
- Deng, X., Lin, C.D., Liu, K.W., Rowe, R.K., 2017. Additive Gaussian process for computer models with qualitative and quantitative factors. *Technometrics* 59, 283–292.
- Deshwal, A., Ament, S., Balandat, M., Bakshy, E., Doppa, J.R., Eriksson, D., 2023. Bayesian optimization over high-dimensional combinatorial spaces via dictionary-based embeddings, in: *International Conference on Artificial Intelligence and Statistics*, PMLR. pp. 7021–7039.
- Deshwal, A., Belakaria, S., Doppa, J.R., 2021a. Bayesian optimization over hybrid spaces, in: *International Conference on Machine Learning*, PMLR. pp. 2632–2643.
- Deshwal, A., Belakaria, S., Doppa, J.R., 2021b. Mercer features for efficient combinatorial bayesian optimization, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7210–7218.
- Deshwal, A., Doppa, J., 2021. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Advances in neural information processing systems* 34, 8185–8200.
- Deville, Y., Ginsbourger, D., Contributors, O.R., Durrande, N., Roustant, M.O., Rcpp, D., DiceKriging, S., Imports, M., Rcpp, L., 2024. Package ‘kergp’. <https://cran.r-project.org/web/packages/kergp/index.html>.
- Dolan, E.D., Moré, J.J., 2002. Benchmarking optimization software with performance profiles. *Mathematical programming* 91, 201–213.
- Dreżkowski, K., Grosnit, A., Bou Ammar, H., 2024. Framework and benchmarks for combinatorial and mixed-variable bayesian optimization. *Advances in Neural Information Processing Systems* 36.
- Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., Wilson, A.G., 2018. Gpytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. *Advances in neural information processing systems* 31.
- Garrido-Merchán, E.C., Hernández-Lobato, D., 2020. Dealing with categorical and integer-valued variables in bayesian optimization with Gaussian processes. *Neurocomputing* 380, 20–35.
- Girard, A., 2004. Approximate methods for propagation of uncertainty with Gaussian process models. University of Glasgow (United Kingdom).
- Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A., 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* 4, 268–276.
- Gopakumar, S., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., 2018. Algorithmic assurance: An active approach to algorithmic testing using bayesian optimisation. *Advances in Neural Information Processing Systems* 31.
- Gower, J.C., 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27, 857–871.
- GPy, since 2012. GPy: A Gaussian process framework in python. <http://github.com/SheffieldML/GPy>.
- Gramacy, R.B., 2020. Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. Chapman and Hall/CRC.
- Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A., 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 723–773.
- Grosnit, A., Malherbe, C., Tutunov, R., Wan, X., Wang, J., Ammar, H.B., 2022. Boils: Bayesian optimisation for logic synthesis, in: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE. pp. 1193–1196.
- Halstrup, M., 2016. Black-box optimization of mixed discrete-continuous optimization problems. Ph.D. thesis, TU Dortmund.
- Jesus, T., Sohst, M., Vale, J.L.d., Suleman, A., 2021. Surrogate based mdo of a canard configuration aircraft. *Structural and Multidisciplinary Optimization* 64, 3747–3771.
- Jones, D.R., Schonlau, M., Welch, W.J., 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 455–492.
- Kabalan, A., Casenave, F., Bordeu, F., Ehrlacher, V., 2025. O-mmgp: Optimal mesh morphing Gaussian process regression for solving pdes with non-parametric geometric variations. *arXiv preprint arXiv:2502.11632*.
- Karlsson, R., Bliet, L., Verwer, S., de Weerd, M., 2020. Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems, in: *Benelux Conference on Artificial Intelligence*, Springer. pp. 48–63.
- Katz, M.H., 2011. Multivariable analysis: a practical guide for clinicians and public health researchers. Cambridge university press.
- Kennedy, M.C., O’Hagan, A., 2001. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 425–464.

- Khan, A., Cowen-Rivers, A.I., Grosnit, A., Robert, P.A., Greiff, V., Smorodina, E., Rawat, P., Akbar, R., Dreczkowski, K., Tutunov, R., et al., 2023. Toward real-world automated antibody design with combinatorial bayesian optimization. *Cell Reports Methods* 3.
- Khuri, A.I., Good, I., 1989. The parameterization of orthogonal matrices: A review mainly for statisticians. *South African Statistical Journal* 23, 231–250.
- Kim, J., Choi, S., Cho, M., 2022. Combinatorial bayesian optimization with random mapping functions to convex polytopes, in: *Uncertainty in Artificial Intelligence*, PMLR. pp. 1001–1011.
- Kirchhoff, D., Kuhnt, S., 2020. Gaussian process models with low-rank correlation matrices for both continuous and categorical inputs. *arXiv preprint arXiv:2010.02574*.
- Kochanski, G., Golovin, D., Karro, J., Solnik, B., Moitra, S., Sculley, D., 2017. Bayesian optimization for a better dessert, in: *NIPS, workshop on Bayesian optimization*.
- Kondor, R.I., Lafferty, J., 2002. Diffusion kernels on graphs and other discrete structures, in: *Proceedings of the 19th international conference on machine learning*, pp. 315–322.
- Krige, D.G., 1951. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy* 52, 119–139.
- Lauvernet, C., Helbert, C., 2020. Metamodeling methods that incorporate qualitative variables for improved design of vegetative filter strips. *Reliability Engineering & System Safety* 204, 107083.
- Liu, D.C., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming* 45, 503–528.
- Maus, N., Jones, H., Moore, J., Kusner, M.J., Bradshaw, J., Gardner, J., 2022. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems* 35, 34505–34518.
- McMillan, N.J., Sacks, J., Welch, W.J., Gao, F., 1999. Analysis of protein activity data by Gaussian stochastic process models. *Journal of Biopharmaceutical Statistics* 9, 145–160.
- Meunier, D., Pontil, M., Ciliberto, C., 2022. Distribution regression with sliced wasserstein kernels, in: *International Conference on Machine Learning*, PMLR. pp. 15501–15523.
- Morris, M.D., Mitchell, T.J., Ylvisaker, D., 1993. Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics* 35, 243–255.
- Moss, H., Leslie, D., Beck, D., Gonzalez, J., Rayson, P., 2020. Boss: Bayesian optimization over string spaces. *Advances in neural information processing systems* 33, 15476–15486.
- Moulavi, D., Jaskowiak, P.A., Campello, R.J., Zimek, A., Sander, J., 2014. Density-based clustering validation, in: *Proceedings of the 2014 SIAM international conference on data mining*, SIAM. pp. 839–847.
- Murtagh, F., Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 2, 86–97.
- Nguyen, D., Gupta, S., Rana, S., Shilton, A., Venkatesh, S., 2020. Bayesian optimization for categorical and category-specific continuous inputs, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5256–5263.
- Notin, P., Hernández-Lobato, J.M., Gal, Y., 2021. Improving black-box optimization in vae latent space using decoder uncertainty. *Advances in Neural Information Processing Systems* 34, 802–814.
- Oh, C., Gavves, E., Welling, M., 2021. Mixed variable bayesian optimization with frequency modulated kernels, in: *Uncertainty in Artificial Intelligence*, PMLR. pp. 950–960.
- Oh, C., Tomczak, J., Gavves, E., Welling, M., 2019. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems* 32.
- Oune, N., Bostanabad, R., 2021. Latent map Gaussian processes for mixed variable metamodeling. *Computer Methods in Applied Mechanics and Engineering* 387, 114128.
- Pelamatti, J., 2020. Mixed-variable Bayesian optimization: application to aerospace system design. Ph.D. thesis. Université de Lille.
- Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.G., Guerin, Y., 2021. Mixed variable Gaussian process-based surrogate modeling techniques: Application to aerospace design. *Journal of Aerospace Information Systems* 18, 813–837.
- Peyré, G., Cuturi, M., et al., 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning* 11, 355–607.
- Phillips, J.M., Venkatasubramanian, S., 2011. A gentle introduction to the kernel distance. *arXiv preprint arXiv:1103.1625*.
- Picheny, V., Wagner, T., Ginsbourger, D., 2013. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and multidisciplinary optimization* 48, 607–626.
- Pinheiro, J.C., Bates, D.M., 1996. Unconstrained parametrizations for variance-covariance matrices. *Statistics and computing* 6, 289–296.
- Qian, P.Z., 2012. Sliced latin hypercube designs. *Journal of the American Statistical Association* 107, 393–399.
- Qian, P.Z.G., Wu, H., Wu, C.J., 2008. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics* 50, 383–396.
- Rapisarda, F., Brigo, D., Mercurio, F., 2007. Parameterizing correlations: a geometric interpretation. *IMA Journal of Management Mathematics* 18, 55–73.
- Rasmussen, C.E., 2003. Gaussian processes in machine learning, in: *Summer school on machine learning*. Springer, pp. 63–71.
- Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20, 53–65.
- Roustant, O., Padonou, E., Deville, Y., Clément, A., Perrin, G., Giorla, J., Wynn, H., 2020. Group kernels for Gaussian process metamodels with categorical inputs. *SIAM/ASA Journal on Uncertainty Quantification* 8, 775–806.
- Ru, B., Alvi, A., Nguyen, V., Osborne, M.A., Roberts, S., 2020. Bayesian optimisation over multiple continuous and categorical inputs, in: *International Conference on Machine Learning*, PMLR. pp. 8276–8285.
- Sack, J., Welch, W., Mitchell, T., Wynn, H., 1989. Design and analysis of computer experiments (with discussion). *Statistical Science* 4, 409–435.
- Santner, T.J., Williams, B.J., Notz, W.I., Williams, B.J., 2003. The design and analysis of computer experiments. volume 1. Springer.
- Saves, P., 2024. High-dimensional multidisciplinary design optimization for aircraft eco-design. Ph.D. thesis. ISAE-SUPAERO.

- Saves, P., Bartoli, N., Diouane, Y., Lefebvre, T., Morlier, J., David, C., Nguyen Van, E., Defoort, S., 2022. Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design, in: AIAA SciTech 2022 Forum, p. 0082.
- Saves, P., Diouane, Y., Bartoli, N., Lefebvre, T., Morlier, J., 2023. A mixed-categorical correlation kernel for Gaussian process. *Neurocomputing* 550, 126472.
- Saves, P., Lafage, R., Bartoli, N., Diouane, Y., Bussemaker, J., Lefebvre, T., Hwang, J.T., Morlier, J., Martins, J.R., 2024. Smt 2.0: A surrogate modeling toolbox with a focus on hierarchical and mixed variables Gaussian processes. *Advances in Engineering Software* 188, 103571.
- Shepard, R., Brozell, S.R., Gidofalvi, G., 2015. The representation and parametrization of orthogonal matrices. *The Journal of Physical Chemistry A* 119, 7924–7939.
- Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* 25.
- Stein, M., 1987. Large sample properties of simulations using latin hypercube sampling. *Technometrics* 29, 143–151.
- Timoshenko, S., Gere, J., 1997. *Mechanics of materials*. PWS 912, 9.
- Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17, 261–272. doi:10.1038/s41592-019-0686-2.
- Wan, X., Nguyen, V., Ha, H., Ru, B., Lu, C., Osborne, M.A., 2021. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. *arXiv preprint arXiv:2102.07188*.
- Watanabe, S., 2023. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*.
- Wistuba, M., Rawat, A., Pedapati, T., 2019. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*.
- Yin, J., Du, X., 2022. Uncertainty quantification by convolutional neural network Gaussian process regression with image and numerical data, in: AIAA SCITECH 2022 Forum, p. 1100.
- Zhang, Y., Notz, W.I., 2015. Computer experiments with qualitative and quantitative variables: A review and reexamination. *Quality Engineering* 27, 2–13.
- Zhang, Y., Tao, S., Chen, W., Apley, D.W., 2020. A latent variable approach to Gaussian process modeling with qualitative and quantitative factors. *Technometrics* 62, 291–302.
- Zhou, Q., Qian, P.Z., Zhou, S., 2011. A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics* 53, 266–273.