Electronic Journal of Statistics
Vol. 18 (2024) 3247–3273
ISSN: 1935-7524
https://doi.org/10.1214/24-EJS2251

A distance metric-based space-filling subsampling method for nonparametric models

Huaimin Diao[†1], **Dianpeng Wang**[†1] **and Xu He**[*2]

[1]*School of Mathematics and Statistics, Beijing Institute of Technology,*
e-mail: 18810190811@163.com; wdp@bit.edu.cn

[2]*MADIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences,*
e-mail: hexu@amss.ac.cn

Abstract: Taking subset samples from the original data set is an efficient and popular strategy to handle massive data that is too large to be directly modeled. To optimize inference and prediction accuracy, it is crucial to employ a subsampling scheme to collect observations intelligently. In this paper, we propose a space-filling subsampling method that uses distance metric-based strata to select subsamples from high-volume data sets. To minimize the maximal distance from pairs of samples that locate in the same stratum, Voronoi cells of thinnest covering lattices are used to partition the input space. In addition, subsamples that are space-filling according to the response are collected from each stratum. With the help of an algorithm to quickly identify the cell an observation locates in, the computational cost of our subsampling method is proportional to the number of observations and irrelevant to the number of cells, which makes our method applicable to extremely large data sets. Results from simulated studies and real data analysis show that the new method is remarkably better than existing approaches when used in conjunction with Gaussian process models.

MSC2020 subject classifications: Primary 60K35, 60K35; secondary 60K35.
Keywords and phrases: Big data, nonparametric model, space-filling design, tall data.

Received February 2023.

1. Introduction

In recent years, high-volume data has found various applications in many scientific and engineering fields. Even though computational resources have proliferated with the rapid development of technology, the size of data grows faster than computational capacity and thus it becomes challenging to restore and model big data sets. An efficient strategy to overcome such difficulty is subsampling, that is, taking a subset of the original massive data and fitting models based on it.

[*]Corresponding author: Xu He.
[†]Co-first author: Huaimin Diao, Dianpeng Wang.

While in most real applications and theoretical advances such as [13] and [24] the observations are selected independently and randomly with equal probability to the subset, referred to as IR hereinafter, more complex sampling methods have shown promising performance in reducing modeling error. Most of such techniques are proposed for linear models or their extensions. For instance, [7] proposed a local case-control subsampling method for imbalanced data sets. [6] proposed a subsampling method based on empirical statistical leverage scores of the covariate matrix to fit linear regression models for large data sets. [19] developed an optimal subsampling procedure based on A- or L-optimality for logistic regression models, which is further extended to softmax regression models by [25]. [21] proposed an information-based optimal subsampling method named IBOSS to select the most informative points based on the D-optimality criterion, which is later improved by a divide-and-conquer strategy [20]. [1] studied the optimal subsampling method under the A-optimality criterion called OSMAC for generalized linear models. Finally, [22] proposed an orthogonal subsampling approach, referred to as OSS hereinafter, for linear regression models.

Although linear models are useful in many applications that are associated with a large number of factors, in many other problems, the data set is tall, i.e., the number of observations, denoted as N hereinafter, is huge while the number of factors, denoted as p hereinafter, is much smaller than N. In such scenarios, linear models are likely found inadequate because they cannot capture complex relations between covariates and the response that can be easily learned from nonparametric models. Subsampling is especially essential in these cases because the time consumption for nonparametric models usually grows rapidly as N increases.

Given the importance of subsampling for nonparametric models, it is somewhat surprising that few methods have been proposed for big tall data sets. Three notable exceptions are [14], [26], and [18]. [14] proposed the support point method, referred to as SP hereinafter, which sample representative points from massive observations such that the distribution of these representative points is close to the distribution of original observations. While this method can be applied to very general applications, it is not tailored for the model fitting purpose. [26] proposed a model-free subsampling method called global likelihood subsampling, referred to as GLSS hereinafter, which allocates massive data into balls and then selects the points from the balls. One shortcoming of the method lies in that many original samples are neglected from the procedure since the union of balls does not cover the whole input space. [18] proposed radial distance metric-based uniform subsampling method, referred to as RU hereinafter, which uses the radial distance metric to cluster massive observations into layers, samples observations uniformly from the layers, and iteratively switches samples from the same layer to minimize the standardized mean squared difference between the quantiles of the subsampled response and the original massive response. Although it has been shown that models fitted from RU subsets are more precise than that from IR, there lacks justification on why samples that have equal distance to the center of the covariates, which can be far away from each other, should be grouped to the same layer. Moreover, RU is slower in com-

putation than fitting Gaussian process model (GP), k-nearest neighbor method (KNN), and many other nonparametric models.

In this paper, we propose a Euclidean distance metric-based space-filling subsampling method, referred to as SF hereinafter, for general nonparametric models. Similar to GLSS and RU, we first divide the original input space into layers and then collect representative points from each layer. However, in order to fit accurate nonparametric models, we propose to group observations close to each other in input values to the same layer and collect representative points from each layer according to their response values.

Take two popular nonparametric models, KNN and GP, to see why we do this. For the k-nearest neighbor method (KNN) [8], the prediction at input location $\boldsymbol{x}$ is the average of the k outcomes corresponding to the k samples that are nearest to $\boldsymbol{x}$. Clearly, the best way to ensure accurate prediction is to collect a certain number of subsamples in any local area. Certainly, each location $\boldsymbol{x}$ belongs to a layer, say w, provided that the layers partition the input space. Let $q(w)$ denote the maximal distance between two points inside the region w, $q(w) = \sup_{\boldsymbol{x}_1,\boldsymbol{x}_2\in w} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$, where $\|\boldsymbol{x}\|$ denotes the Euclidean distance from $\boldsymbol{x}$ to the zero vector. If k or more subsamples are collected from w, it is guaranteed that there are at least k subsamples that are located within distance $q(w)$ to $\boldsymbol{x}$. Because the bias of the prediction is related to $q(w)$, it is desirable to partition the input space into layers with low q values, which requires to group samples close to each other under the Euclidean distance measure to the same layer. In addition, it is desirable that the subsamples are representative of the layer. Remark that if the subsamples are purely randomly collected, there is a chance that the responses are all low or all high relative to the majority of response values within the layer, causing variance error on predictions. To reduce this variance error as much as possible, when we decide to select z subsamples from the layer w, we partition the samples within w into z almost equally-sized sublayers by grouping samples with close response values together and then select the sample with the median response value from each sublayer.

Gaussian process model (GP) is another type of widely-used nonparametric method. [23] provides an upper bound on the modeling error, which is approximately proportional to $\sigma h(\boldsymbol{x})^\rho$, where σ and ρ are positive constants related to the variance and smoothness of the response function and $h(\boldsymbol{x})$ is the distance between $\boldsymbol{x}$ and its nearest sample. Clearly, it is desirable that our subsamples have low $h(\boldsymbol{x})$ for any $\boldsymbol{x}$. In fact, space-filling designs that minimize the maximum of $h(\boldsymbol{x})$, also referred to as minimax distance designs [11], are popular for GP when the experiments are designed. As long as we collect at least one sample from the layer w that $\boldsymbol{x}$ belongs to, the $h(\boldsymbol{x})$ will not exceed $q(w)$ and thus the modeling error is controlled.

From the above arguments, it is appealing to select space-filling subsamples and one practical solution is to minimize the maximal q of the layers. The use of response values aside from input values to select subsamples further reduces random error. From our method, we minimize the maximal q by using Voronoi cells of the thinnest covering lattice as the layers, whose definitions and details will be provided in Section 2.2. From numerical results that are based on KNN

and GP, our newly proposed subsampling method SF performs well compared with other subsampling methods.

The remainder of this paper is organized as follows. In Section 2, the new subsampling strategy is proposed and analyzed. In Sections 3 and 4, simulations and real data examples are provided to demonstrate the advantage of our proposed subsampling method. Several conclusions and remarks are given in Section 5. Proofs and some detailed derivations are provided in Appendix.

2. Subsampling methodology

2.1. Major steps

Our subsampling method can be seen as a proportionate stratified sampling method that proceeds with three major parts below:

I. Partition the input space into a list of cells.
II. Classify the samples to the cells.
III. Sample representative points from each cell parallelly and collect the selected samples.

In subsections to follow, we give details on the three major parts of our method. Throughout this paper, let N, N_w, n, n_w, m, and p denote the number of original samples, the number of original samples that locate in cell w, the number of selected samples, the number of selected samples from cell w, the number of cells, and the number of covariates, respectively.

2.2. Rotated lattice-based partitioning

In this subsection, we provide Part I of our method, i.e., the steps to partition the input space into cells. To begin with, Figure 1 gives two partitionings of $[0,1]^2$, a square partitioning and a hexagonal partitioning. In both partitionings, the cells are Voronoi cells of a lattice. A full-rank lattice $\boldsymbol{L} \subset \mathbb{R}^p$ is an infinite set of points that can be expressed as the linear combination of p linearly independent p-dimensional vectors with integer coefficients [3]. That is, $\boldsymbol{L}$ can be expressed as $\{\boldsymbol{a}^*\boldsymbol{G} : \boldsymbol{a}^* \in \mathbb{Z}^p\}$ with a full rank $p \times p$ matrix $\boldsymbol{G}$, where $\mathbb{Z}^p$ denotes the set consisting of p-dimensional integer vectors. The Voronoi cell of a lattice point $\boldsymbol{c}_i$ that belongs to a lattice $\boldsymbol{L}$ is the region that is nearest to it than other lattice points,

$$\mathrm{Vor}(\boldsymbol{c}_i) = \{\boldsymbol{z} : \|\boldsymbol{z} - \boldsymbol{c}_i\| = \min_{\boldsymbol{c}\in\boldsymbol{L}} \|\boldsymbol{z} - \boldsymbol{c}\|\}.$$

Clearly, the Voronoi cells of a lattice are identical and the input space is a subset of the union of the Voronoi cells. Furthermore, if the layers are given by Voronoi cells of a lattice, the maximal q is twice the radius of the Voronoi cells. In this sense, the lattice points jointly cover the input space using radius $q/2$ balls centered at them. For instance, the set of p-dimensional integer vectors $\{\boldsymbol{a}^*\boldsymbol{I}_p : \boldsymbol{a}^* \in \mathbb{Z}^p\}$ is a lattice, where $\boldsymbol{I}_p$ denotes the p-dimensional identity

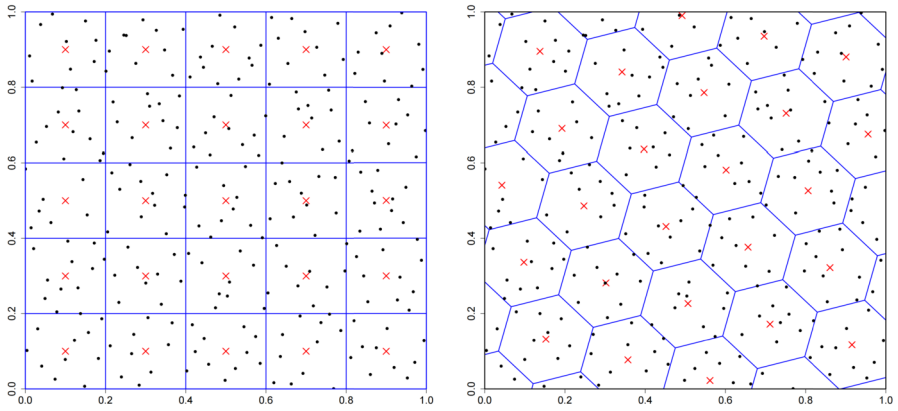

FIG 1. *Square (left) and hexagonal (right) partitioning of* $[0,1]^2$*, showing the cells (blue squares or hexagons), centers (red crosses), and original massive samples (black dots).*

matrix, and the cells in the left panel of Figure 1 are Voronoi cells of this lattice. Let m denote the number of cells. From simple derivations, we find that the radius of this square partitioning is roughly $(2m)^{-1/2}$ when m is large, and q is two times the radius, i.e., q is asymptotically $2^{1/2}m^{-1/2} = 1.4142m^{-1/2}$ for all squares. In contrast, from the regular hexagon partitioning illustrated in the right panel of Figure 1, q is asymptotically $2^{3/2}3^{-3/4}m^{-1/2} = 1.2408m^{-1/2}$, which is smaller than that for the square partitioning. We thus consider the hexagon partitioning to be more space-filling than the square partitioning.

We can control the maximal q by using the Voronoi cells of a space-filling type of lattice as the layers. The type of p-dimensional lattice that has the lowest asymptotic q is called the p-dimensional thinnest covering lattice. For $2 \le p \le 22$, these lattices can be expressed as [3]

$$\boldsymbol{L} = \{l^*\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R} : \boldsymbol{a}^* \in \mathbb{Z}^p\}, \tag{1}$$

where l^* is a positive constant, $\boldsymbol{M}^* = \boldsymbol{I}_p - [\{(1+p)^{1/2}+1+p\}/\{p(p+1)\}]\boldsymbol{J}_{p,p}$, $\boldsymbol{J}_{p,p}$ denotes the $p \times p$ matrix with all elements being one, and $\boldsymbol{R}$ is a p-dimensional orthogonal matrix. Here $\{\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R} : \boldsymbol{a}^* \in \mathbb{Z}^p\}$ is a rotation of the lattice $\{\boldsymbol{a}^*\boldsymbol{M}^* : \boldsymbol{a}^* \in \mathbb{Z}^p\}$ since $\boldsymbol{R}$ is orthogonal and $\{l^*\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R} : \boldsymbol{a}^* \in \mathbb{Z}^p\}$ is a rescaling of $\{\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R} : \boldsymbol{a}^* \in \mathbb{Z}^p\}$. Because rotation and rescaling do not change the shape of Voronoi cells, the type of lattice and its asymptotic q property are solely determined by $\boldsymbol{M}^*$. From $\boldsymbol{L}$ in (1), q is asymptotically $(p+1)^{(1-p)/(2p)}\{p(p+2)/3\}^{1/2}m^{-1/p}$ [3]. In contrast, for regular grids $\{l\boldsymbol{a}\boldsymbol{I}_p : \boldsymbol{a} \in \mathbb{Z}^p\}$, which are also called integer lattices, q is asymptotically $p^{1/2}m^{-1/p}$. Table 1 summarizes the constants corresponding to $m^{-1/p}$ for the two types of lattices in $2 \le p \le 10$. It can be seen that $\boldsymbol{L}$ in (1) has substantially smaller constants than regular grids. For $p > 22$ the $\boldsymbol{L}$ in (1) is not the thinnest covering lattice. However, since in this paper, we focus on low p problems, it suffices to consider this type of lattice.

TABLE 1
The constants corresponding to $m^{-1/p}$ of thinnest covering lattices (TCL) in (1) and integer lattices (IL).

p	2	3	4	5	6	7	8	9	10
TCL	1.24	1.41	1.55	1.67	1.78	1.88	1.97	2.06	2.15
IL	1.41	1.73	2.00	2.24	2.45	2.65	2.83	3.00	3.16

Although asymptotic q value remains unchanged from rotating the lattice, as discussed in [9] and [10], rotating the lattice makes cell centers to be more uniformly distributed after projected onto a subset of dimensions of $\{1, \ldots, p\}$, which is appealing when there is sparsity among input variables. We will verify this point in Section 3.4. We thus propose to use Voronoi cells of rotated thinnest covering lattices as the layers.

By definition, a lattice contains infinitely many points. However, the number of centers of cells intercepting the input space is finite. Let $\boldsymbol{\Omega} \subset \mathbb{R}^p$ denote an input space with volume V and $\boldsymbol{\Omega} \oplus \boldsymbol{\epsilon} = \{\boldsymbol{\xi} + \boldsymbol{\epsilon} : \boldsymbol{\xi} \in \boldsymbol{\Omega}\}$ denote a translation of $\boldsymbol{\Omega}$. For $\boldsymbol{L}$ in (1) and uniformly distributed random $\boldsymbol{\epsilon}$, the expected number of elements of $\boldsymbol{L} \cap (\boldsymbol{\Omega} \oplus \boldsymbol{\epsilon})$ is $l^{*-p}(p+1)^{1/2}V$ [9]. Consequently, by using $l^* = m^{-1/p}(p+1)^{1/(2p)}V^{1/p}$, there will be roughly m cell centers.

2.3. An efficient algorithm to identify the cell

In Part II of the method, we classify observations into cells. Because a cell consists of the region that is closest to its center than other centers, the classification problem becomes a problem to identify the closest center to observations. Obviously, this can be done by computing the distance between each observation and each center. The complexity of computing the distance from one center to one original observation is $\mathcal{O}(p)$, the complexity of identifying the nearest center from m centers to one original observation is $\mathcal{O}(mp)$, and thus the complexity of completing the above identification algorithm is $\mathcal{O}(Nmp)$.

To apply our subsampling methodology to scenarios with huge N and large m, a more efficient algorithm is essential. Remark that the problem of finding the closest lattice center to a given point is the well-known closest vector problem, which is known to be NP-complete for arbitrary lattices [5]. Nevertheless, from exploiting properties of the type of lattice we use, namely, the $\boldsymbol{L}$ in (1), in this subsection, we propose a classification algorithm that requires $\mathcal{O}\{Np^2 \log(p)\}$ computations. This is much faster than $\mathcal{O}(Nmp)$ since p is small in our assumption and, as we shall show in later sections, the best choice of m seems to be proportional to n and thus $\mathcal{O}(Nmp)$ is big. The ease of quickly finding the closest lattice center is another reason besides low q of using thinnest covering lattice-based partitioning rather than other types of lattice or non-lattice partitions.

Let $[\lambda]$ denote the integer closest to λ, $\boldsymbol{v}_\tau$ denote the p-vector with the first τ entries being 1 and the rest entries being 0, $\boldsymbol{u}_\tau = l^* \boldsymbol{v}_\tau \boldsymbol{M}^* \boldsymbol{R}$,

$$\tilde{\boldsymbol{R}} = \begin{pmatrix} \boldsymbol{R} & 0\boldsymbol{J}_{p,1} \\ 0\boldsymbol{J}_{1,p} & 1 \end{pmatrix},$$

$$\boldsymbol{F} = (p+1)^{-1/2} \begin{pmatrix} \boldsymbol{J}_{p,1} & (p+1)^{1/2}\boldsymbol{I}_p - p^{-1}\{(p+1)^{1/2}+1\}\boldsymbol{J}_{p,p} \\ 1 & \boldsymbol{J}_{p,1}^T \end{pmatrix},$$

$\tilde{\boldsymbol{v}}_\tau = (\boldsymbol{v}_\tau, 0)$, $\tilde{\boldsymbol{u}}_\tau = (\boldsymbol{u}_\tau, 0)$, $\boldsymbol{U} = \tilde{\boldsymbol{R}}^T \boldsymbol{F}/l^*$, and $\boldsymbol{U}_\tau = \tilde{\boldsymbol{u}}_\tau \boldsymbol{U}$. Algorithm 1 below gives the steps to find the index of the closest center of an observation $\boldsymbol{x}$. Remark that in the algorithm we do not compute the coordinates of cell centers. Instead, we compute indices of centers given by $(\hat{\tau}, \tilde{\boldsymbol{s}}_{\hat{\tau}})$. We have three reasons of doing so. Firstly, as long as we can group original observations that belong to the same cell together, there is no need to know the coordinates of centers. Secondly, obtaining the coordinates requires substantially more computations. Thirdly, the $(\hat{\tau}, \tilde{\boldsymbol{s}}_{\hat{\tau}})$'s are more convenient in coding because they are integer vectors.

Algorithm 1: The procedure to identify the closest center to given observation $\boldsymbol{x}$

1 Compute $\tilde{\boldsymbol{y}} = (\boldsymbol{x}, 0)\boldsymbol{U}$.
2 **for** $\tau = 0, \ldots, p$ **do**
3 Compute $\boldsymbol{s}_\tau = \tilde{\boldsymbol{y}} - \boldsymbol{U}_\tau$, $\bar{\boldsymbol{s}}_\tau = [\boldsymbol{s}_\tau]$, $\boldsymbol{\delta}_\tau = \boldsymbol{s}_\tau - \bar{\boldsymbol{s}}_\tau$, and $\Delta_\tau = \bar{\boldsymbol{s}}_\tau \boldsymbol{J}_{p+1,1}$.
4 Let $\boldsymbol{\beta}_\tau$ be the $(p+1)$-vector consisting of zeros.
5 If $\Delta_\tau > 0$, replace the entries of $\boldsymbol{\beta}_\tau$ that are corresponding to the Δ_τ lowest entries of $\boldsymbol{\delta}_\tau$ by "-1".
6 If $\Delta_\tau < 0$, replace the entries of $\boldsymbol{\beta}_\tau$ that are corresponding to the $-\Delta_\tau$ highest entries of $\boldsymbol{\delta}_\tau$ by "1".
7 Compute $\tilde{\boldsymbol{s}}_\tau = \bar{\boldsymbol{s}}_\tau + \boldsymbol{\beta}_\tau$ and $\|\boldsymbol{\beta}_\tau - \boldsymbol{\delta}_\tau\|$.
8 Obtain $\hat{\tau}$, the τ that minimizes $\|\boldsymbol{\beta}_\tau - \boldsymbol{\delta}_\tau\|$.
9 Output $(\hat{\tau}, \tilde{\boldsymbol{s}}_{\hat{\tau}})$, an index of the cell that is closest to $\boldsymbol{x}$.

Note that Steps 5 and 6 in the algorithm are well-defined only if $\Delta_\tau \in \mathbb{Z}$ and $|\Delta_\tau| \leq (p+1)/2$. Proposition 2.1 below verifies that this is always the case.

Proposition 2.1. *From Algorithm 1, $\Delta_\tau \in \mathbb{Z}$ and $|\Delta_\tau| \leq (p+1)/2$ for any $\tau \in \{0, \ldots, p\}$.*

Theorem 2.2 below assures that Algorithm 1 identifies the correct center.

Theorem 2.2. *Suppose $(\hat{\tau}, \tilde{\boldsymbol{s}}_{\hat{\tau}})$ is the output from Algorithm 1 with input $\boldsymbol{x}$. Let $\tilde{\boldsymbol{z}}_\tau(\boldsymbol{x}) = l^* \tilde{\boldsymbol{s}}_\tau \boldsymbol{F}^T \tilde{\boldsymbol{R}} + \tilde{\boldsymbol{u}}_\tau$ and $\boldsymbol{z}_\tau(\boldsymbol{x})$ denote the first p entries of $\tilde{\boldsymbol{z}}_\tau(\boldsymbol{x})$. Then the last entry of $\tilde{\boldsymbol{z}}_\tau(\boldsymbol{x})$ is zero and $\boldsymbol{z}_{\hat{\tau}}(\boldsymbol{x})$ gives the center among lattice centers $\boldsymbol{L}$ in (1) that is closest to $\boldsymbol{x}$.*

The proofs of Proposition 2.1 and Theorem 2.2 are provided in Appendix. The intuition of the algorithm is as follows: Firstly, the thinnest covering lattice $\boldsymbol{L}$ in (1) can be partitioned into $p+1$ slices via $\boldsymbol{L} = \bigcup_{\tau=0}^{p} (\boldsymbol{K} \oplus \boldsymbol{u}_\tau)$, where $\boldsymbol{K}$

denotes the lattice given by $\{l^*\boldsymbol{bMR} : \boldsymbol{b} \in \mathbb{Z}^p\}$, $\boldsymbol{Q}$ is the $p \times p$ matrix defined by

$$\boldsymbol{Q} = \begin{pmatrix} 2 & 1 & 1 & \cdots & 1 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix},$$

$\boldsymbol{M}$ is the $p \times p$ matrix defined by

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{QM}^* \\ &= \boldsymbol{I}_p - \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} - \frac{(p+1)^{\frac{1}{2}}+1}{p} \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix}, \end{aligned}$$

and $\boldsymbol{K} \oplus \boldsymbol{u}_\tau = \{\boldsymbol{z} + \boldsymbol{u}_\tau : \boldsymbol{z} \in \boldsymbol{K}\}$ is a translation of $\boldsymbol{K}$. Secondly, from multiplying $\boldsymbol{U}$ any point in $\mathbb{R} \times \{0\}$ is rotated to a hyperplane of $\mathbb{R}^{p+1}$ such that the lattice centers of $\boldsymbol{K} \oplus \boldsymbol{u}_\tau$ are mapped into integer vectors. Exploiting this specific structure of centers, [3] has provided an algorithm that requires $\mathcal{O}\{p \log(p)\}$ computations to find the lattice center of $\boldsymbol{K} \oplus \boldsymbol{u}_\tau$ that is nearest to any given point in the hyperplane. In light of these results, we identify the closest lattice center from each slice and then find out which of the $p+1$ centers are closest to $\boldsymbol{x}$.

2.4. Details of the algorithm and complexity analysis

Let

$$\begin{aligned} \boldsymbol{R}_1 &= \begin{pmatrix} (1-2^{\frac{1}{2}})\{20-14(2^{\frac{1}{2}})\}^{-\frac{1}{2}} & (1+2^{\frac{1}{2}})\{20+14(2^{\frac{1}{2}})\}^{-\frac{1}{2}} \\ \{3-2(2^{\frac{1}{2}})\}\{20-14(2^{\frac{1}{2}})\}^{-\frac{1}{2}} & \{3+2(2^{\frac{1}{2}})\}\{20+14(2^{\frac{1}{2}})\}^{-\frac{1}{2}} \end{pmatrix}, \\ \boldsymbol{R}_2 &= \begin{pmatrix} (1-5^{\frac{1}{2}})\{20-8(5^{\frac{1}{2}})\}^{-\frac{1}{2}} & (1+5^{\frac{1}{2}})\{20+8(5^{\frac{1}{2}})\}^{-\frac{1}{2}} \\ (3-5^{\frac{1}{2}})\{20-8(5^{\frac{1}{2}})\}^{-\frac{1}{2}} & (3+5^{\frac{1}{2}})\{20+8(5^{\frac{1}{2}})\}^{-\frac{1}{2}} \end{pmatrix}, \\ \boldsymbol{R}_3 &= \begin{pmatrix} (1-13^{\frac{1}{2}})\{52-12(3^{\frac{1}{2}})\}^{-\frac{1}{2}} & (1+13^{\frac{1}{2}})\{52+12(3^{\frac{1}{2}})\}^{-\frac{1}{2}} \\ (5-13^{\frac{1}{2}})\{52-12(3^{\frac{1}{2}})\}^{-\frac{1}{2}} & (5+13^{\frac{1}{2}})\{52+12(3^{\frac{1}{2}})\}^{-\frac{1}{2}} \end{pmatrix}, \end{aligned}$$

and $\boldsymbol{A} \otimes \boldsymbol{B}$ denote the Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Algorithm 2 below give the detailed steps to select n samples from the training data set $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, N\}$.

We have several comments for Algorithm 2. Firstly, when $p \in \{2, 4, 8\}$, we use the recommended fixed rotation matrix $\boldsymbol{R}$ in [10]. Otherwise, we use random $\boldsymbol{R}$ as recommended by [9] because no fixed rotation matrix for such p was provided

Algorithm 2: The main SF subsampling procedure

Input: p, $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, N\}$, n, m, and $\hat{V}$.
Output: A set giving the n selected samples.

1 **if** $p = 2$ **then**
2 Let $\boldsymbol{R} = \boldsymbol{I}_p$.
3 **else if** $p = 4$ **then**
4 Let $\boldsymbol{R} = \boldsymbol{R}_1 \otimes \boldsymbol{R}_2$.
5 **else if** $p = 8$ **then**
6 Let $\boldsymbol{R} = \boldsymbol{R}_1 \otimes \boldsymbol{R}_2 \otimes \boldsymbol{R}_3$.
7 **else**
8 **for** $i = 1, \ldots, p-1$ *and* $j = i+1, \ldots, p$ **do**
9 Let $\boldsymbol{R}_{i,j}(\alpha_{i,j})$ be the identify matrix with the (i,i)-th, (i,j)-th, (j,i)-th, and (j,j)-th entries be replaced by $\cos(\alpha_{i,j})$, $-\sin(\alpha_{i,j})$, $\sin(\alpha_{i,j})$, and $\cos(\alpha_{i,j})$, respectively, where $\alpha_{i,j}$ is randomly sampled from the uniform distribution on $[0, 2\pi]$.
10 Let $\boldsymbol{R} = \prod_{1 \le i < j \le p} \boldsymbol{R}_{i,j}(\alpha_{i,j})$.
11 Compute $l^* = m^{-1/p}(p+1)^{1/(2p)}\hat{V}^{1/p}$, $\boldsymbol{F}$, $\boldsymbol{U} = \tilde{\boldsymbol{R}}^T\boldsymbol{F}/l^*$, $\boldsymbol{M}^*$, $\tilde{\boldsymbol{u}}_\tau$, and $\boldsymbol{U}_\tau = \tilde{\boldsymbol{u}}_\tau\boldsymbol{U}$.
12 **for** $i = 1, \ldots, N$ **do**
13 Identify the cell that $\boldsymbol{x}_i$ lies in by using Algorithm 1.
14 Calculate the number of actual cells, denoted as $\hat{m}$.
15 For each $w \in \{1, \ldots, \hat{m}\}$, calculate the number of points lie in cell w, denoted as N_w, and let $n_w = \lceil nN_w/N \rceil$.
16 **while** $\sum_w (n_w) > n$ **do**
17 Find the w that yields the highest $n_w - nN_w/N$ among those with $n_w > 1$ and let $n_w = n_w - 1$.
18 For $w = 1, \ldots, \hat{m}$, collect the n_w points closest to the $(1-1/2)/n_w$, $(2-1/2)/n_w, \ldots, (n_w - 1/2)/n_w$ quantiles of the response from the cell w.
19 Output the n collected samples.

in [10]. These matrices are by far the best choices. Secondly, the algorithm requires a prior estimate of V, which is denoted as $\hat{V}$. If the estimate is poor, the actual m we find after Step 14, which is denoted as $\hat{m}$, may be far away from the m we aim to achieve. In scenarios that we do not have confidence in the $\hat{V}$ used, we recommend to check if the $\hat{V}$ is reasonable after identifying 10% of original samples during Step 13. Let $\bar{m}$ and $\bar{m}_1$ denote the numbers of cells that contain at least one sample and exactly one sample, respectively, after identifying 10% samples. Then an estimate of $\hat{m}$ is $\bar{m} + 9\bar{m}_1/10$ and an re-estimate of V is $\tilde{V} = \hat{V}(\bar{m} + 9\bar{m}_1/10)/m$. Details of our derivation on these estimators are provided in the appendix. If the re-estimate is significantly different from $\hat{V}$, we should adjust our estimate on V using the above rule and restart Algorithm 2 from Step 11. Nevertheless, the estimate of V needs not to be very accurate because from simulation results we shall show in Sections 3 and 4, SF performs robustly well when $m \in [0.3n, 0.7n]$. For the same reason, we recommend using $m = n/2$. Thirdly, from Steps 15–17 the subsample sizes are determined to be proportional to the large data in each cell. However, we ensure that each cell has at least one subsample. Finally, in Step 18 we sort and partition the samples within

the w layer into n_w almost equally-sized sublayers by their response values and then select the most representative sample from each sublayer, i.e. the one with the median response value. This maximizes the space-filling property of the response value for the select representative samples from each cell in the sense that the empirical distribution of the selected samples is closest to the empirical distribution of the whole samples in the cell. Also because the cells are space-filling for the input variables, the final subsamples are space-filling for both inputs and outputs.

In the rest of this section, we derive the complexity of SF. From Algorithm 1, the computational complexity of Steps 3, 4, 5, 6, and 7 are $\mathcal{O}(p)$, $\mathcal{O}(p)$, $\mathcal{O}\{p\log(p)\}$, $\mathcal{O}\{p\log(p)\}$, and $\mathcal{O}(p)$, respectively. Therefore, in sum, the computational complexity of Steps 2–8 is $\mathcal{O}\{p^2\log(p)\}$. Also because the computational complexity of Step 1 and Step 9 is $\mathcal{O}(p^2)$, altogether the computational complexity of identifying the cell of one center is $\mathcal{O}\{p^2\log(p)\}$. Consequently, the computational complexity of Steps 12–13 of Algorithm 2 is $\mathcal{O}\{Np^2\log(p)\}$. On the other hand, provided that proper data structure is employed so that covariate and response values of samples identified to each cell can be quickly collected, in Step 18 of Algorithm 2, it requires $\mathcal{O}\{N_w\log(N_w)\}$ computations to select the representative points for group w. Summing these complexity, in total $\mathcal{O}\{N\log(N)\}$ computations is required. Because other steps of Algorithm 2 require less computation, in sum SF requires no more than $\mathcal{O}\{Np^2\log(p) + N\log(N)\}$ computations. More often than not, $N\log(N)$ is smaller than $Np^2\log(p)$, rendering the time complexity to be $\mathcal{O}\{Np^2\log(p)\}$. Because the cost is largely proportional to $p^2\log(p)$, SF is not applicable to large p problems. On the other hand, because the cost is proportional to N and irrelevant to n or m, SF is applicable to problems with huge N and n. In sum, SF is appropriate for tall data sets.

As a comparison, the computational complexity of RU is $\mathcal{O}\{\Lambda Nn\log(n) + N\log(N)\}$, where Λ is a parameter in the RU algorithm that determines the number of iterations to switch representative samples in each stratum [18]. In fact, the actual Λ from default settings is almost always greater than n. Assuming that Λ is proportional to n, the computational complexity of RU is $\mathcal{O}\{Nn^2\log(n) + N\log(N)\}$, which is much larger than SF for tall data sets. On the other hand, the computational complexity of SP is $\mathcal{O}\{\Lambda Nnp\}$, where Λ is the number of iterations [14]. From computation code provided in [14], the Λ by default is in between of 50 and 250, which is not related to n. Consequently, SP is in general faster than RU but slower than SF. Finally, the computational complexity of OSS is $\mathcal{O}\{Np\log(n)\}$, and the computational complexity of GLSS is $\mathcal{O}\{Np\log(N)+M\log(N)\}$, where M is the size of the generated low-discrepancy point set [26]. Empirically, SF is faster than OSS and GLSS when $p \leq 4$ but becomes slower than OSS and GLSS when $p \geq 5$.

In practice, the prediction accuracy of nonparametric models is likely to be more sensitive to the number of training samples n than the uniformity of training samples. Consequently, it is not wise to invest a major part of computational resource on subsampling rather than on model fitting and prediction. For KNN, the computational complexity of making one prediction is $\mathcal{O}(np)$. Usually, a

cross-validation procedure is employed to identify the optimal k for KNN. If altogether H choices of k are tried and N_t testing samples are used to validate each choice of k, it requires $\mathcal{O}(npHN_t)$ computations, which is larger than $\mathcal{O}\{Np^2\log(p) + N\log(N)\}$ in many scenarios. For GP, the computational complexity of fitting one model with fixed roughness and scale parameters is $\mathcal{O}(n^3)$ [17], which is much larger than $\mathcal{O}\{Np^2\log(p) + N\log(N)\}$ in most scenarios. Moreover, the scale parameters from GP are usually estimated from data, and the parameter estimation procedure requires much larger computation time than fitting one GP model. Therefore, we conclude that the SF method is suitable to be used in combination with KNN or GP models.

There are two techniques to further reduce the computation of SF. Firstly, in many applications, some covariates are remarkably more effective in predicting the response than other covariates and these important covariates can be found without too much complexity. In these cases, we can use the important covariates only for the SF algorithm. Since the computational cost of SF is mainly proportional to $p^2\log(p)$, by reducing p, the cost of SF can be remarkably reduced. Secondly, we can select $\hat{N}$ samples from the original data set using IR and then apply SF to select n samples from the $\hat{N}$ samples. This strategy applies to applications with very big N.

3. Simulation study

3.1. Comparison setup

In this section, we compare our proposed subsampling method to several other types of methods using five numerical examples. The methods in the comparison include:

- **IR** Pure random subsampling in which observations are selected independently with equal probability.
- **SP** The support point subsampling method by [14].
- **RU** The proportionate stratified subsampling method that uses the radial distance metric to partition the original observations [18].
- **OSS** The orthogonal subsampling method by [22].
- **GLSS** The global likelihood subsampling method by [26].
- **SF-m** Our newly proposed thinnest covering lattice-based space-filling subsampling method that partitions the input space into m Voronoi cells.
- **HC-m** The space-filling subsampling method that is the same as SF-m but uses hypercubes (squares) instead of Voronoi cells of the thinnest covering lattices to partition the input space.

For each testing function $g(\boldsymbol{x})$ that will be provided in subsections to follow, we generate $N = 10^4$ or $N = 10^5$ observations as the training data set. We then use the seven subsampling methods to obtain a subset of $n = 1{,}000$ training samples and fit KNN and GP models using them. For SF, the number of cells m is set to be 100, 300, 500, or 700 while m is set to be 500 for HC. We evaluate

the prediction accuracy using the mean absolute error (MAE) given by

$$MAE = N_t^{-1} \sum_{i=1}^{N_t} |g(\boldsymbol{x}_i) - \hat{g}(\boldsymbol{x}_i)|, \tag{2}$$

where N_t is the size of testing data and $g(\boldsymbol{x}_i)$ and $\hat{g}(\boldsymbol{x}_i)$ denote the real and fitted response value from the i-th testing sample, respectively. We repeat the data generation, subsampling, and model-fitting process 100 times and record all MAE values. Because RU is very time-consuming for these examples, we set the number of iterations to be 50, which is lower than the default value, to accelerate the computation.

3.2. A 2-dimensional example with uniformly distributed observations

We use the Branin-Hoo function [16] below as the first test function:

$$\begin{aligned} g(\boldsymbol{x}) = 1/51.95[\{\bar{x}_2 - 5.1\bar{x}_1^2/(4\pi^2) + 5\bar{x}_1/\pi\}^2 \\ + \{10 - 10/(8\pi)\}\cos(\bar{x}_1) - 44.81], \end{aligned}$$

where $\bar{x}_1 = 15x_1 - 5$, $\bar{x}_2 = 15x_2$, and in the full data set $\boldsymbol{x} = (x_1, x_2)$ is assumed to follow the uniform distribution on $[0, 1]^2$.

Figure 2 shows the boxplots of MAE values from the subsampling methods except OSS. Here we omit the results for OSS because the MAE for OSS is dramatically higher than that for other subsampling methods. In fact, all subsampling points that OSS selects are in the corners, which is appealing for linear models but certainly undesirable for KNN or GP. From the results, SF-500 and SF-700 are the best methods for GP. For KNN, SP is the best method and SF-500 is the second best. However, the MAEs for KNN are thousands of times higher than those for GP, showing that SF-500 or SF-700 with GP is the best combination. In all cases, HC-500 is only slightly inferior to SF-500. This shows that the choice of the type of covering does not contribute much to the performance of subsampling methods. We thus infer that our strategies of using Voronoi cells to partition the input space and select subsamples for each cell according to response values play a more important role in reducing MAE.

Using one 3.2-GHz core of our computer, it takes roughly 0.04 minutes, 0.02 minutes, 0.50 minutes, 0.12 minutes, 0.12 minutes, and 19.88 minutes for SF, HC, SP, OSS, GLSS, and RU to select one subset of $N = 10^4$ samples, respectively, while it takes roughly 0.64 minutes, 0.51 minutes, 4.94 minutes, 1.13 minutes, 1.23 minutes, and 864.57 minutes for SF, HC, SP, OSS, GLSS and RU to select one subset of $N = 10^5$ samples, respectively. On the other hand, it takes roughly 2.69 minutes to fit one GP model. As the time consumed in SF is negligible compared to that for GP modeling, we conclude that SF is appealing when GP is employed. In contrast, even though we accelerate the algorithm by modifying the default number of iterations, RU still takes much longer time than

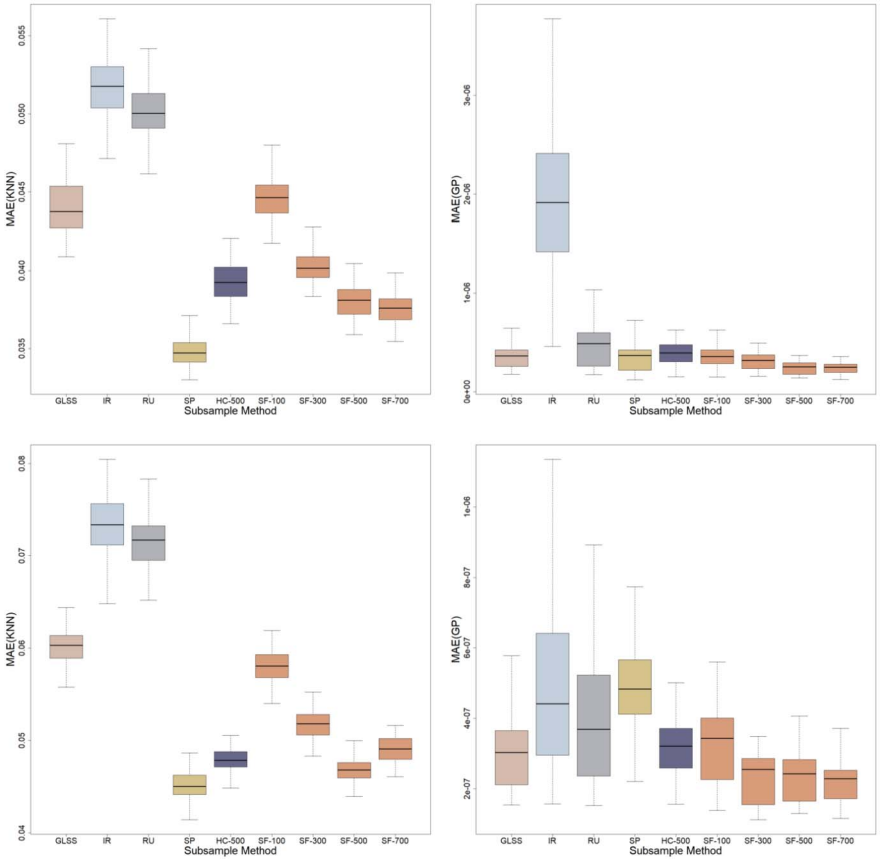

FIG 2. *MAE values from the KNN (left) and GP (right) models for the Branin-Hoo test function when N is 10^4 (top) and 10^5 (bottom) where the covariates follow the uniform distribution on $[0,1]^2$.*

fitting GP model, showing that RU is not appropriate for the problem unless a model that requires much longer fitting time than GP is employed. The SP, OSS, and GLSS are also slower than SF and HC. Fitting a KNN model usually requires a cross-validation procedure to find the optimal k, namely the number of neighbors that is employed in prediction. Provided that altogether 10^4 and 10^5 validation samples are used to compare 20 choices of k's, it takes 1.5 seconds and 21.67 seconds to find the optimal k, respectively. Although KNN requires much less time, it is substantially not as accurate as GP in prediction.

3.3. A 2-dimensional example with Gaussian distributed observations

In our next example, we use the same Branin-Hoo function that was used in the previous subsection. However, we assume for the full data set $\boldsymbol{x} = (\varphi_1, \varphi_2 +$

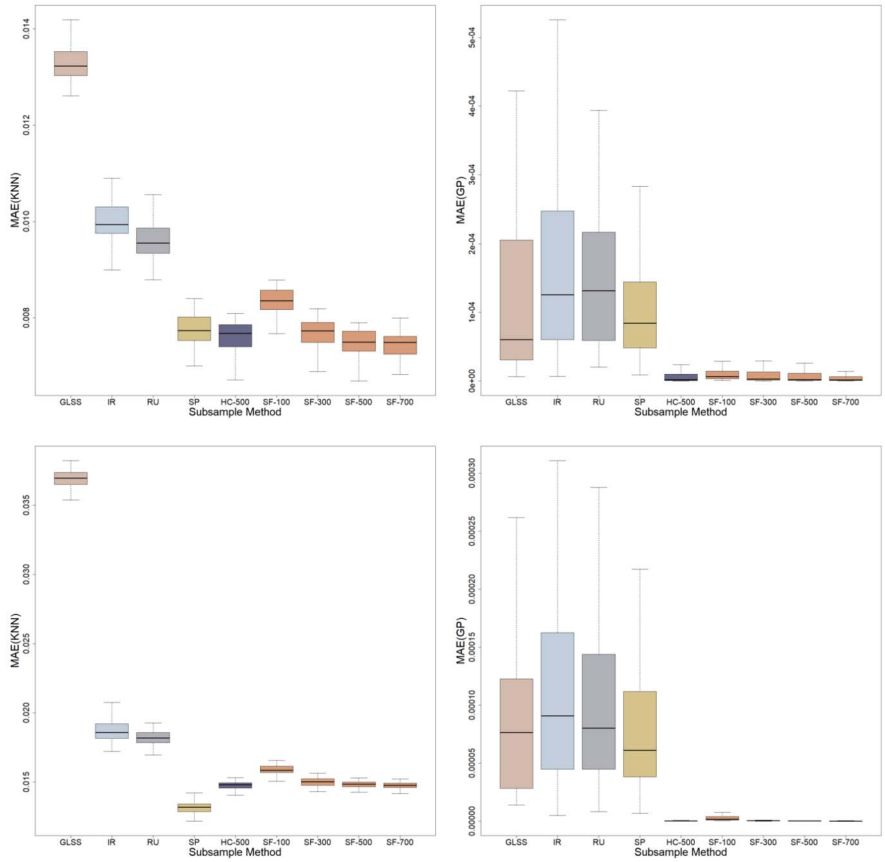

FIG 3. *MAE values from the KNN (left) and GP (right) models for the Branin-Hoo test function when N is 10^4 (top) and 10^5 (bottom) where the covariates follow a bivariate Gaussian distribution.*

$\varphi_1^2/1.2$) and (φ_1, φ_2) follows the standard bivariate Gaussian distribution.

Figure 3 shows the boxplots of MAE of the subsampling methods. Again, we omit the results for OSS because they are very poor. The comparison is similar to that in Section 3.2, except that the gap in performance between SF/HC and other methods is even wider for GP. This suggests that SF and HC are more advantageous when the inputs are not uniformly distributed. Furthermore, the MAE values are insensitive to m, especially for GP, showing that SF is robust on the choice of m. Since the N, p, and n are the same, it takes roughly the same computation time as in the previous example.

3.4. A 2-dimensional example with sparsity on input variables

Next, we use the

$$g(\boldsymbol{x}) = \cos(55x_1^2),$$

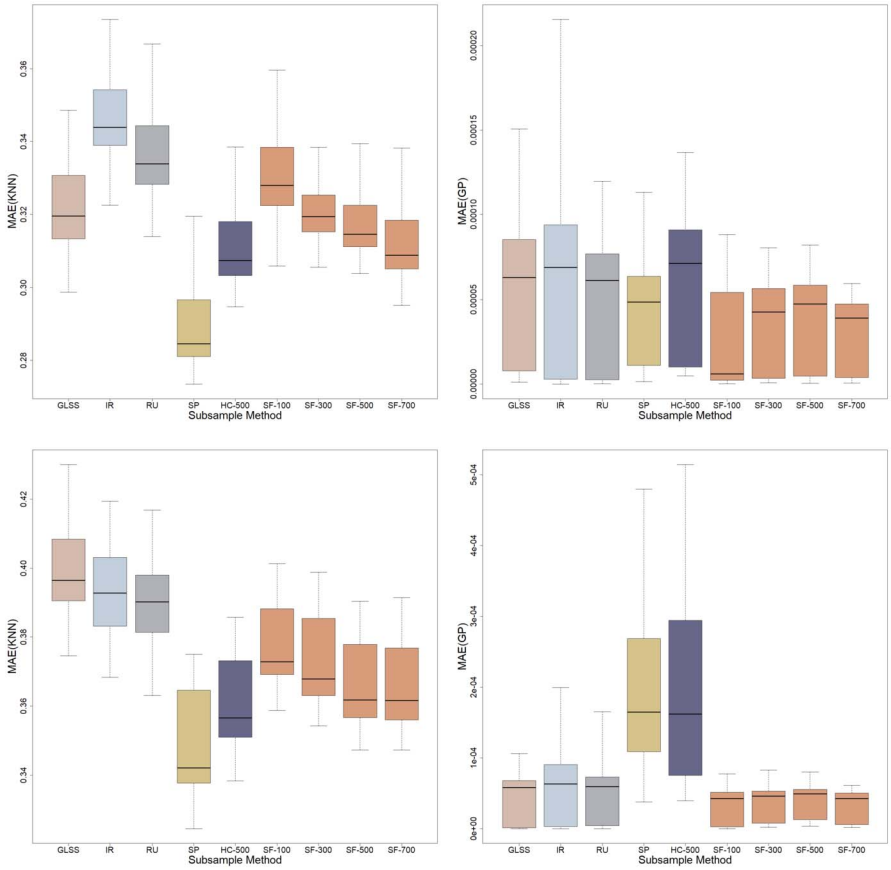

FIG 4. *MAE values from the KNN (left) and GP (right) models when N is 10^4 (top) and 10^5 (bottom), where the covariates follow the uniform distribution on $[0,1]^2$ and there is an inert covariate.*

where $\boldsymbol{x} = (x_1, x_2)$ is assumed to follow the uniform distribution on $[0,1]^2$ as our third test function. Clearly, for this function, only x_1 is related to the output value. However, we assume that the data set contains two input variables and we do not know that x_2 does not affect the output when we select subsamples.

Figure 4 shows the boxplots of MAE values from the subsampling methods. Again, we omit the results for OSS because they are very poor. The comparison is again similar to that provided in Section 3.2. However, we find that for this function HC-500 is remarkably inferior to SF-500 for GP when $N = 10^5$. To explain this observation, in Figure 5, we show the histograms of the x_1 of the subsamples selected from HC-500 and SF-500 from one repetition. We can see from the histograms that the subsamples from SF-500 are much more uniform than those from HC-500. This is because the cell centers of the rotated thinnest covering lattice are more uniform when projected onto one dimension than that from the unrotated integer lattice. As only one variable is important, from the

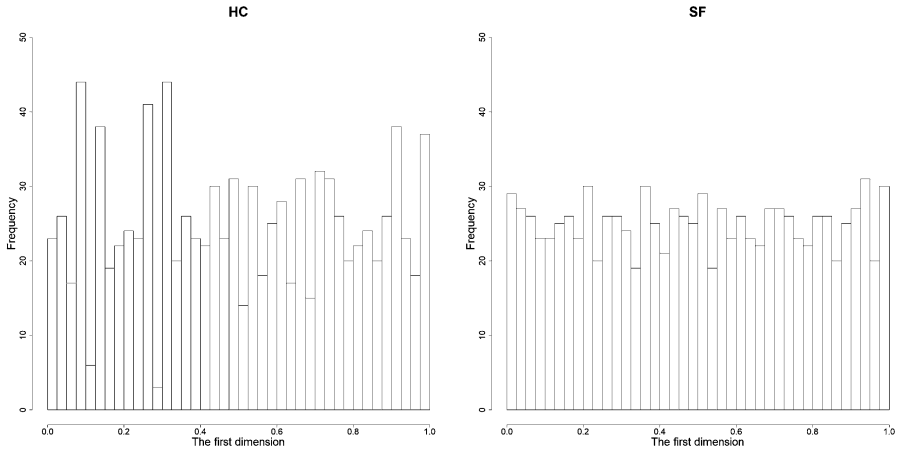

FIG 5. *Histograms of the first input variable of the subsample obtained by HC-500 (left) and SF-500 (right) from the* $N = 10^5$ *training data.*

GP model the prediction is almost solely determined by x_1. It is thus desirable that the Voronoi cells are space-filling after being projected onto the first dimension. Moreover, since the five squares in each column of the graph in the left panel of Figure 1 will be projected to the same interval, the cells become much less space-filling after the projection. On the other hand, since the projections of the hexagons of the right panel in Figure 1 do not coincide, the hexagons are more space-filling than the squares after the project. We conjecture that this makes SF-500 better than HC-500. To sum it up, when there is sparsity among input variables subsampling method that is based on more uniformly projected cells is better.

3.5. A 3-dimensional example

Next, we use the 3-dimensional function below, which is similar to the function used in [12], as our fourth test function:

$$g(\boldsymbol{x}) = \sin(\pi(x_1 + x_2 + x_3)/3) - x_1 - x_2^2,$$

where $\boldsymbol{x} = (x_1, x_2, x_3)$ follows the uniform distribution on $[0, 1]^3$.

Figure 6 shows the boxplots of MAE values for the subsampling methods. The comparison is again similar to that provided in Section 3.2. For this example, it takes roughly 0.05, 0.02, 0.66, 0.13, 0.13, and 22.79 minutes for SF, HC, SP, OSS, GLSS, and RU to select one subsample from $N = 10^4$ samples, while it takes roughly 0.75, 0.63, 5.92, 1.31, 1.36, and 997.02 minutes for SF, HC, SP, OSS, GLSS, and RU to select one subset from $N = 10^5$ samples. On the other hand, it takes roughly 3.68 minutes to fit one GP model. Again, the time consumed by SF subsampling is negligible compared with GP modeling, showing that SF can be applied when GP is used.

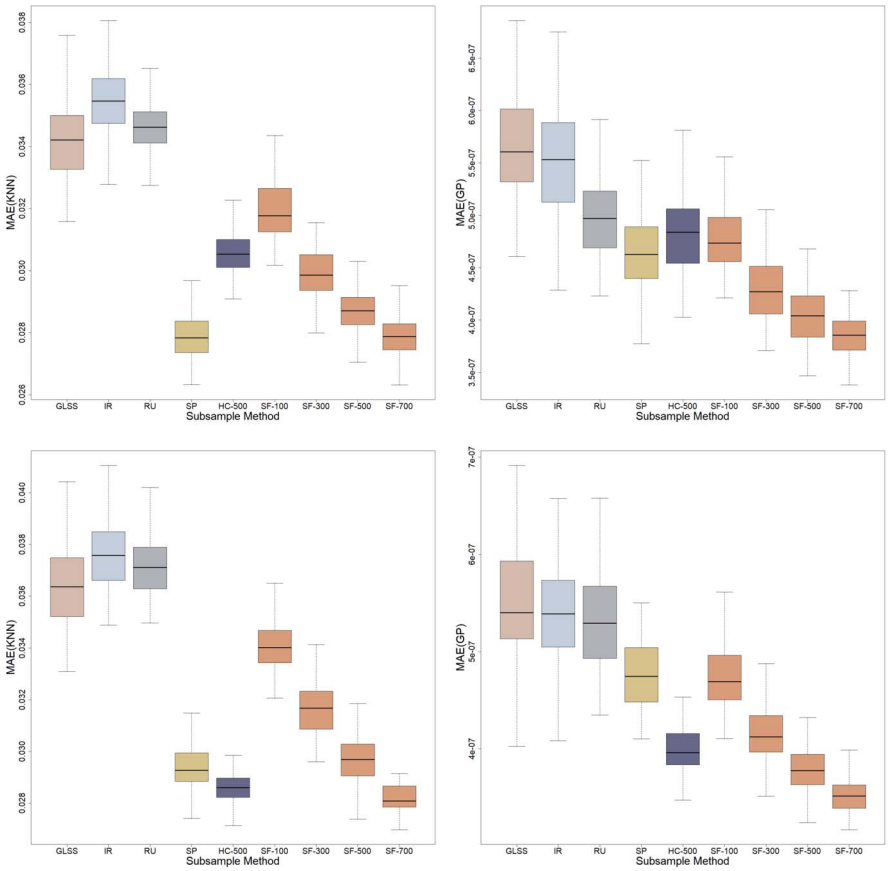

FIG 6. *MAE values from the KNN (left) and GP (right) models when N is 10^4 (top) and 10^5 (bottom) where the covariates follow the uniform distribution on $[0,1]^3$.*

3.6. A 6-dimensional example

Finally, we use the 6-dimensional spiky function below [2] as our fifth test function:

$$g(\boldsymbol{x}) = 10^4/2[\phi\{10(\boldsymbol{x} - 1/3)\} + \phi\{10(\boldsymbol{x} - 2/3)\}],$$

where $\phi(\boldsymbol{\kappa}) = \{1/(2\pi)\}^2 \exp(-0.5\|\boldsymbol{\kappa}\|^2)$ and $\boldsymbol{x}$ follows the uniform distribution on $[0,1]^6$.

Figure 7 shows boxplots of MAE values for the subsampling methods. Unlikely the first four test functions, for this function OSS performs the best. For both KNN and GP, SF and HC are the second-best methods and are considerably better than IR, SP, GLSS, and RU. Furthermore, SF with larger m is slightly better than SF with lower m. For this example, it takes roughly 0.15, 0.02, 0.99, 0.13, 0.14 and 23.8 minutes for SF, HC, SP, OSS, GLSS, and RU to

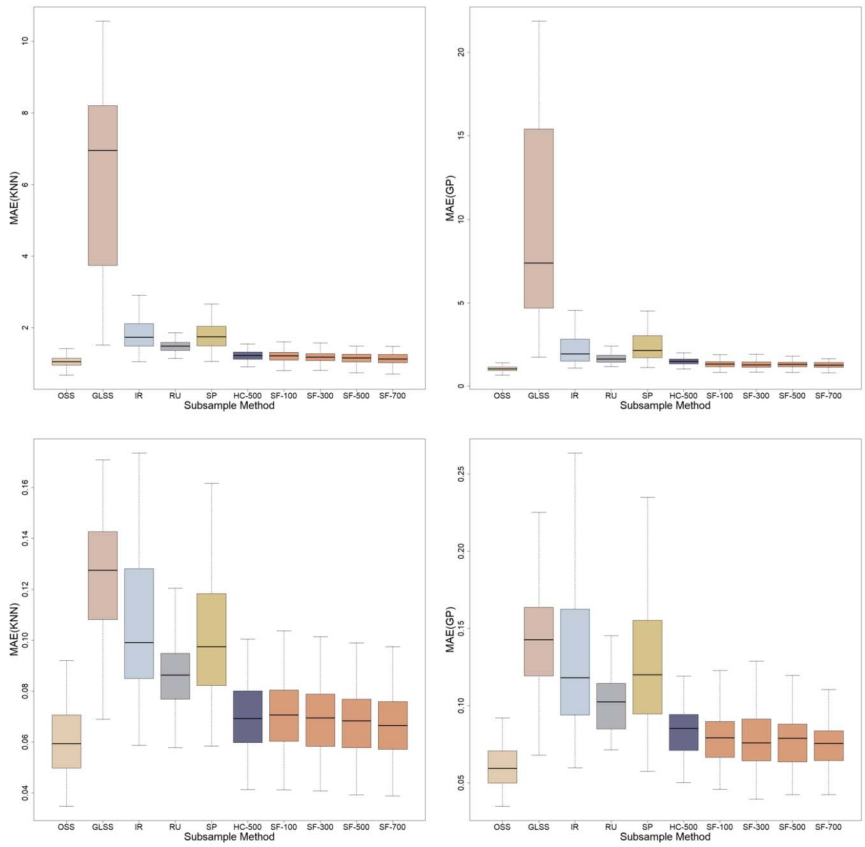

FIG 7. *MAE values from the KNN (left) and GP (right) models when N is 10^4 (top) and 10^5 (bottom) where the covariates follow the uniform distribution on $[0,1]^6$.*

select one subsample for $N = 10^4$, while it takes roughly 2.07, 0.68, 9.81, 1.31, 1.51, and 1012.53 minutes for SF, HC, SP, OSS, GLSS, and RU to select one subsample for $N = 10^5$. Meanwhile, it takes roughly 3.76 minutes to fit one GP model. Recall that the computational complexity of SF is $\mathcal{O}\{Np^2\log(p)\}$. This is why for this 6-dimensional example, SF takes much longer time than previous 2-dimensional and 3-dimensional examples. Nevertheless, SF is still faster than fitting a GP model, showing that SF is applicable for GP modeling.

We remark that in all five examples, the N and n are smaller than that of real-world big data problems. We do not set larger N or n values here because, for numerical comparison purposes, we need to repeat the subsampling and modeling procedure 100 times. Provided that replication is not needed, SF is applicable to much bigger data sets because its time consumption is largely linear to N and not related to n. Also note that the code we use for SF is written in R. Provided that we rewrite our code in C, we expect SF to run much faster.

4. Real data examples

4.1. Total column ozone data

We continue to compare subsampling methods using real data sets. The first data set is the level 2 total column ozone data, which was calibrated and pre-processed by NASA and previously studied by [4] and [15]. The data set has one response, the ozone content, and two covariates giving the longitude and latitude of the position.

The original data set contains 173,405 observations. We randomly choose $N_t = 34{,}681$ observations as testing samples to evaluate the accuracy of predictions, and randomly select $\hat{N} = 10{,}000$ observations from the remaining $N = 138,724$ observations as the candidates for subsampling methods. We then apply the seven subsampling methods to select $n = 500$ or $n = 1000$ subsamples from the $\hat{N} = 10{,}000$ observations and fit KNN and GP models. For SF, we set m to be 50, 150, 250, and 350 when $n = 500$ and 100, 300, 500, and 700 when $n = 1000$. For HC, we set m to be 250 and 500 when $n = 500$ and 1000, respectively. For reliable comparison of subsampling methods, we repeat the whole process of randomly choosing $\hat{N}$ observations, subsampling, and model fitting 100 times. Remark that it is possible for SF to select samples from the original 173,405 observations directly. However, this will be very time-consuming if RU is applied or repetition is introduced.

Figure 8 shows the boxplots of MAE values of the seven subsampling methods excepts OSS. We omit the results for OSS because the MAE for OSS is much higher than that for other methods. Seen from the results, for both $n = 500$ and $n = 1000$ scenarios GP is substantially better than KNN. For GP, SF with $m = n/2$ and $m = 0.7n$ perform the best and HC-500 is only slightly inferior to SF-500. For KNN, SP is the best method and SF-700, SF-500, and HC-500 are the second best methods. The computation time for SF, HC, SP, OSS, GLSS, RU, and GP model fitting are roughly 0.03 minutes, 0.01 minutes, 0.24 minutes, 0.06 minutes, 0.07 minutes, 16.8 minutes, and 0.23 minutes, respectively, when $n = 500$ and 0.04 minutes, 0.02 minutes, 0.50 minutes, 0.12 minutes, 0.12 minutes, 22.28 minutes, and 2.39 minutes, respectively, when $n = 1000$. Similar to numerical examples provided in Section 3, SF and HC are the fastest methods which are much faster than GP model fitting. Furthermore, as the time consumption of GP model fitting is known to be proportional to n^3 while the time consumption of SF is largely unrelated to n, we expect that the ratio of time consumption between SF and GP to be smaller for higher n. This is partially confirmed by comparing the $n = 500$ and $n = 1000$ scenarios here.

4.2. Windfarm data

The second data set is the windfarm data that was used in [18] to evaluate subsampling methods. The data set consists of 31,266 data points with seven covariates, which are generator torque, ramp rate between generator torque and

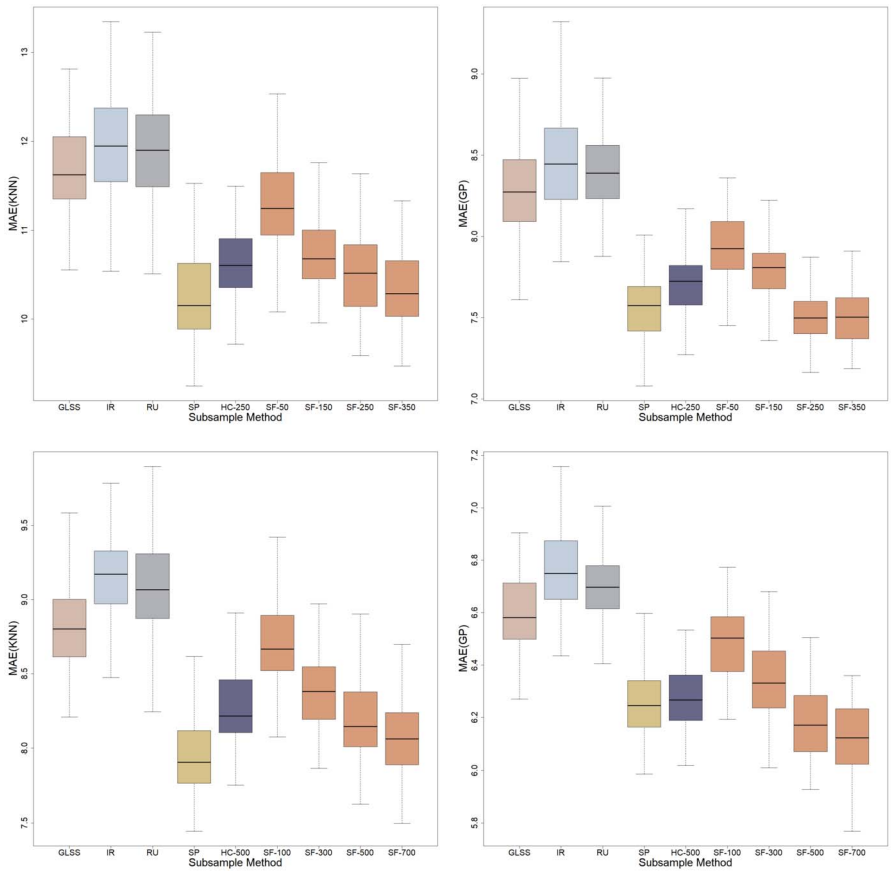

FIG 8. *MAE values from the KNN (left) and GP (right) models when n is 500 (top) and 1000 (bottom) for the total column ozone data.*

blade pitch angle, blade pitch angle, ramp rate between wind speed and blade pitch angle, wind speed, ramp rate between wind speed, and wind deviation of a commercial wind farm. The purpose here is to use the seven covariates to predict the response, namely tower vibratory acceleration. Because the data set is observational, the seven covariates are not uniformly distributed. We set $N_t = 10{,}422$ samples as testing samples and the rest $N = 20{,}844$ observations as training samples. We then apply the seven subsampling methods to select $n = 1{,}000$ subsamples from the $N = 20{,}844$ observations and fit KNN and GP models. The whole process of subsampling and model fitting is repeated 100 times.

Figure 9 gives the MAE values from the seven subsampling methods. Since OSS and GLSS have dramatically larger MAE than the other five methods, which can be ovserved from the top two panels, in the bottom two panels we omit the results from the two methods. From the results, GP is better than

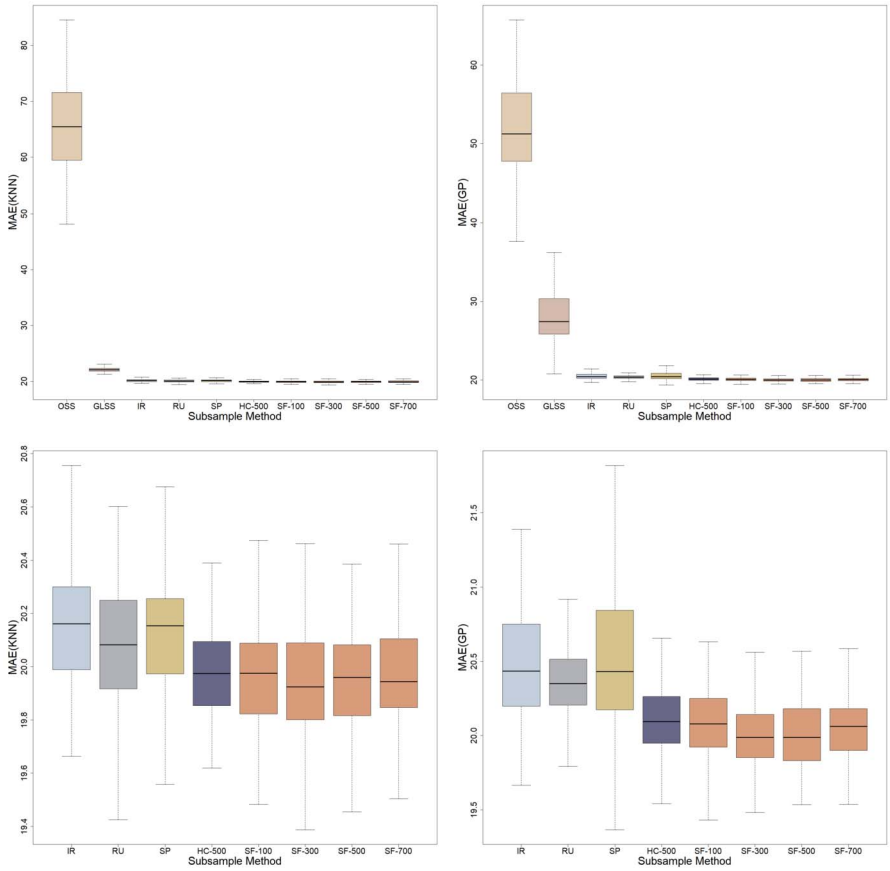

FIG 9. *MAE values from the KNN (left) and GP (right) models for all seven subsampling methods (top) and the five subsampling methods except OSS and GLSS (bottom) for the windfarm data.*

KNN in MAE for all subsampling methods. For both KNN and GP, SF and HC have the best MAE values, while SF-500 is slightly better than HC-500 for GP. Moreover, the performance of SF is insensitive to the choice of m. Remark that for this data set, the testing samples are themselves associated with random errors and thus no method can predict the testing response very well. The time consumption for SF, HC, SP, OSS, GLSS, RU, and GP are roughly 0.3 minutes, 0.13 minutes, 1.13 minutes, 0.19 minutes, 0.3 minutes, 56.54 minutes, and 4.88 minutes, respectively.

Since the time consumption of SF is proportional to $p^2 \log(p)$, one approach to reduce SF time consumption is to reduce the number of covariates considered in the subsampling process. To exploit this idea, we fit a Lasso regression model that consists of the intercept and linear and quadratic main effects and find that four out of the seven covariates contribute 99.48% of the total sensitivity

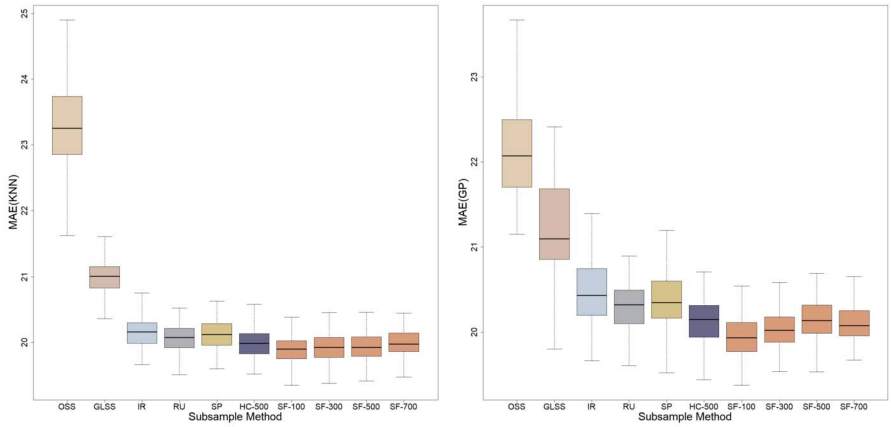

FIG 10. *MAE values from the KNN (left) and GP (right) models for the windfarm data in which the subsampling process is associated with four covariates only.*

indices. We then try using the four covariates only for the subsampling methods, whose results are provided in Figure 10. Remark that in fitting KNN and GP models, we still use all seven covariates. Since the performance of the subsampling methods is almost the same as those when using all seven corvariates in subsampling, we conclude that for windfarm data, it suffices to use four covariates in choosing subsamples. Using four corvariates, it takes SF, HC, SP, OSS, GLSS, and RU roughly 0.11, 0.03, 0.74, 0.17, 0.27, and 55.16 minutes to select one subset of samples. Obviously, SF and HC are faster than fitting GP models, SP, OSS, GLSS, and RU. This cues us that in practice, we may accelerate SF by using important covariates only.

5. Conclusions and discussion

In this paper, we introduce a new subsampling method called SF for large-N-small-p data sets. The SF method can largely be seen as a proportionate stratified sampling method in which the strata are Voronoi cells of thinnest covering lattices. In addition, for each cell SF selects subsamples that are space-filling according to their response values. From numerical results based on simulated and real data sets, the new method is remarkably better than existing approaches when used in conjunction with Gaussian process models. We have also tried a modification of SF that uses hypercubes as the strata. Although this method performs as good as SF under several scenarios, we find that when there is sparsity among corvariates the hypercube subsampling method may not be as good as SF. The SF method is fast in computation when the number of covariates is small. Because the most time-consuming steps are identifying the cell of samples and selecting representative samples from cells and both steps can be run in parallel, we believe the SF method can be accelerated by parallel computation.

The newly proposed subsampling method may be suitable for other nonparametric models, but more work needs to be done to verify its performance. It is also interesting to see if the new method can be used for other computer experiment problems such as numerical integration and quantile estimation. The subsampling method might as well be useful for data checking. For example, if a cell contains only one sample, the sample is a potential outlier; on the other hand, if a cell contains unusually too many samples, it is possibly because the error code of missing values falls into the cell.

Appendix A: Proof of Proposition 2.1

Proof. Because $\boldsymbol{F}\boldsymbol{J}_{p+1,1} = (0,\ldots,0,(p+1)^{1/2})^T$, for any $\boldsymbol{x} \in \mathbb{R}^p$ and $0 \le \tau \le p$,

$$\boldsymbol{s}_\tau \boldsymbol{J}_{p+1,1} = (\boldsymbol{x}-\boldsymbol{u}_\tau, 0)\tilde{\boldsymbol{R}}^T \boldsymbol{F}\boldsymbol{J}_{p+1,1} = ((\boldsymbol{x}-\boldsymbol{u}_\tau)\boldsymbol{R}^T, 0)(0,\ldots,0,(p+1)^{1/2})^T = 0.$$

Because the entries of $\bar{\boldsymbol{s}}_\tau$ and $\boldsymbol{s}_\tau$ differ by at most $1/2$, $|\Delta_\tau| \le (p+1)/2$. Because $\bar{\boldsymbol{s}}_\tau \in \mathbb{Z}^p$, $\Delta_\tau \in \mathbb{Z}$. □

Appendix B: Proof of Theorem 2.2

Proof. Let $\boldsymbol{G}$ be the $p \times p$ matrix given by

$$\boldsymbol{G} = \begin{pmatrix} 1 & -(p-1) & -(p-2) & \cdots & -1 \\ 1 & 2 & -(p-2) & \cdots & -1 \\ 1 & 2 & 3 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & -1 \\ 1 & 2 & 3 & \cdots & p \end{pmatrix} /(p+1).$$

Then from simple derivations $\boldsymbol{M}^* = \boldsymbol{G}\boldsymbol{M}$. Furthermore, for any integer vector $\boldsymbol{a}^* \in \mathbb{Z}^p$, $\boldsymbol{a}^*\boldsymbol{G} \in \mathbb{Z}^p$ if and only if $p+1$ divides $\sum_{i=1}^p a_i^*$. For any $\boldsymbol{x} \in L$, there exists an $\boldsymbol{a}^* \in \mathbb{Z}^p$ such that $\boldsymbol{x} = l^*\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R}$. Clearly, there exists an $0 \le \tau \le p$ such that $(\sum_{i=1}^p a_i^* - \tau)/(p+1) \in \mathbb{Z}$. Let $\boldsymbol{b} = (\boldsymbol{a}^* - \boldsymbol{v}_\tau)\boldsymbol{G}$, then $\boldsymbol{b} \in \mathbb{Z}^p$. Because

$$l^*\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R} = l^*\boldsymbol{v}_\tau\boldsymbol{M}^*\boldsymbol{R} + l^*(\boldsymbol{a}^* - \boldsymbol{v}_\tau)\boldsymbol{G}\boldsymbol{M}\boldsymbol{R} = \boldsymbol{u}_\tau + l^*\boldsymbol{b}\boldsymbol{M}\boldsymbol{R},$$

$\boldsymbol{x} \in \boldsymbol{K} \oplus \boldsymbol{u}_\tau$. Conversely, for any $\boldsymbol{x} \in \boldsymbol{K} \oplus \boldsymbol{u}_\tau$, there is an $\boldsymbol{b} \in \mathbb{Z}^p$ such that $\boldsymbol{x} - \boldsymbol{u}_\tau = l^*\boldsymbol{b}\boldsymbol{M}\boldsymbol{R}$. Let $\boldsymbol{a}^* = \boldsymbol{v}_\tau + \boldsymbol{b}\boldsymbol{Q}$. Then $\boldsymbol{x} = l^*\boldsymbol{a}^*\boldsymbol{M}^*\boldsymbol{R}$, i.e., $\boldsymbol{x} \in \boldsymbol{L}$. To sum it, $\boldsymbol{K} \oplus \boldsymbol{u}_\tau$ with $\tau = 0,\ldots,p$ partitions $\boldsymbol{L}$.

Let $\tilde{\boldsymbol{M}}$ be the $p \times (p+1)$ matrix whose first p columns is $\boldsymbol{M}$ and the last column consists of zeros and

$$\boldsymbol{P} = \tilde{\boldsymbol{M}}\boldsymbol{F} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}.$$

Because $\tilde{\boldsymbol{s}}_\tau \boldsymbol{J}_{p+1,1} = \bar{\boldsymbol{s}}_\tau \boldsymbol{J}_{p+1,1} + \boldsymbol{\beta}_\tau \boldsymbol{J}_{p+1,1}$, there exists an integer $\boldsymbol{c} \in \mathbb{Z}^p$ such that $\tilde{\boldsymbol{s}}_\tau = \boldsymbol{cP}$. Therefore,

$$\tilde{\boldsymbol{z}}_\tau(\boldsymbol{x}) = l^* \tilde{\boldsymbol{s}}_\tau \boldsymbol{F} \tilde{\boldsymbol{R}} + \tilde{\boldsymbol{u}}_\tau = l^* \boldsymbol{cPF}^T \tilde{\boldsymbol{R}} + \tilde{\boldsymbol{u}}_\tau = l^* \boldsymbol{c} \tilde{\boldsymbol{M}} \tilde{\boldsymbol{R}} + \tilde{\boldsymbol{u}}_\tau.$$

Because the first p columns of $\tilde{\boldsymbol{M}}$ is $\boldsymbol{M}$ and the last row with the first p columns of $\tilde{\boldsymbol{R}}$ is $\boldsymbol{R}$, the first p columns of $\tilde{\boldsymbol{M}}\tilde{\boldsymbol{R}}$ is $\boldsymbol{MR}$. Because the last column of $\tilde{\boldsymbol{M}}$ consists of zeros and the first p rows with the last column of $\tilde{\boldsymbol{R}}$ consists with zeros, the last column of $\tilde{\boldsymbol{M}}\tilde{\boldsymbol{R}}$ consists of zeros. Also because $\tilde{\boldsymbol{u}}_\tau = (\boldsymbol{u}_\tau, 0)$, the last entry of $\tilde{\boldsymbol{z}}_\tau(\boldsymbol{x})$ is 0 and

$$\boldsymbol{z}_\tau(\boldsymbol{x}) = l^* \boldsymbol{cMR} + \boldsymbol{u}_\tau \in \boldsymbol{K} \oplus \boldsymbol{u}_\tau.$$

Because $\tilde{\boldsymbol{R}}$ and $\boldsymbol{F}$ are orthogonal matrices, $l^*\boldsymbol{U}$ is an orthogonal matrix. Therefore, for arbitrary $\boldsymbol{c} \in \mathbb{Z}^p$,

$$\begin{aligned} \|(l^* \boldsymbol{cMR} + \boldsymbol{u}_\tau) - \boldsymbol{x}\| &= \|l^* \boldsymbol{c} \tilde{\boldsymbol{M}} \tilde{\boldsymbol{R}} + \tilde{\boldsymbol{u}}_\tau - (\boldsymbol{x}, 0)\| \\ &= l^* \|l^* \boldsymbol{c} \tilde{\boldsymbol{M}} \tilde{\boldsymbol{R}} \boldsymbol{U} + \tilde{\boldsymbol{u}}_\tau \boldsymbol{U} - (\boldsymbol{x}, 0)\boldsymbol{U}\| \\ &= l^* \|\boldsymbol{cP} - \boldsymbol{s}_\tau\|. \end{aligned}$$

Because $\boldsymbol{cPJ}_{1,p+1} = 0$,

$$\|\boldsymbol{cP} - \boldsymbol{s}_\tau\| \geq \|\tilde{\boldsymbol{s}}_\tau - \boldsymbol{s}_\tau\|.$$

That is, $\boldsymbol{z}_\tau(\boldsymbol{x})$ is one of the closest point in $\boldsymbol{K} \oplus \boldsymbol{u}_\tau$ to $\boldsymbol{x}$.

Also because $\boldsymbol{K} \oplus \boldsymbol{u}_\tau$ with $\tau = 0, \ldots, p$ partitions $\boldsymbol{L}$, $\boldsymbol{z}_{\hat{\tau}}(\boldsymbol{x})$ is one of the closest point in $\boldsymbol{L}$ to $\boldsymbol{x}$. □

Appendix C: Estimation of $\boldsymbol{V}$

Proof. Let m_j denote the number of cells that contain exactly j samples after all N original samples are identified to the cells. Then the expected number of cells that contain exactly one sample after $\bar{N} \in \mathbb{N}$ samples are identified, $\bar{m}_1$, is

$$\begin{aligned} \bar{m}_1 &= \sum_{j=1}^{N} \left\{ m_j \bar{N}(j/N) \frac{N-j}{N-1} \cdots \frac{N-j-\bar{N}+2}{N-\bar{N}+1} \right\} \\ &= \sum_{j=1}^{N} \left\{ m_j \bar{N}(j/N) \frac{N-\bar{N}}{N-1} \cdots \frac{N-j-\bar{N}+2}{N-j+1} \right\}. \end{aligned}$$

When $\bar{N}$ is much less than N, it is not hard to see that the terms with $j > 3N/\bar{N}$ do not contribute much to the summation. Also assuming that $m_1 = \cdots = m_{3N/\bar{N}} = x$ and plugging $\bar{N} = N/10$, $\bar{m}_1$ is approximately

$$(x/10) \sum_{j=1}^{+\infty} \left\{ j(9/10)^{j-1} \right\} = 10x.$$

Consequently, $\bar{m}_1/10$ is an estimate of x after $N/10$ samples are identified.

Similarly, the expected number of cells (among those who contain at least one point after all N original samples are identified) that contain no sample after $N/10 \in \mathbb{N}$ samples are identified, $\bar{m}_0$, is approximately

$$x \sum_{j=1}^{N} \frac{N-j}{N} \cdots \frac{N-j-(N/10)+1}{N-(N/10)+1} \sum_{j=1}^{+\infty} (9/10)^j = 9x.$$

Consequently, $9\bar{m}_1/10$ is an estimate of $\bar{m}_0$ after $N/10$ samples are identified. Therefore, an estimate of the number of cells that contain at least one point after all N original samples are identified is $\bar{m} + 9\bar{m}_1/10$, where $\bar{m}$ is the number of cells that contain at least one sample after $N/10$ samples are identified. Consequently, an re-estimate of V is $\hat{V}(\bar{m}+9\bar{m}_1/10)/m$, where m and $\hat{V}$ are the target number of cells and the estimated volume of the input space, respectively, that have been used in identifying the $N/10$ samples. □

Acknowledgments

We thank the editor, one associate editor, and three referees for their constructive comments that lead to significant improvement of the paper.

Funding

This work is supported by National Key R&D Program of China 2021YFA 1000300, 2021YFA 1000301, National Natural Science Foundation of China (Grant no. NSFC 12171033, NSFC 12022115, NSFC 11971465, NSFC 71988101), China Institute of Marine Technology and Economy (Contact Number 2019A128), and National Center for Mathematics and Interdisciplinary Sciences, CAS.

References

[1] Ai, M., J. Yu, H. Zhang, and H. Wang (2021). Optimal subsampling for big data regressions. *Statistica Sinica 31*, 749–772. MR4286193

[2] An, J. and A. Owen (2001). Quasi-regression. *Journal of Complexity 17*(4), 588–607. MR1881660

[3] Conway, J. H. and N. J. A. Sloane (1998). *Sphere Packings, Lattices and Groups*. Springer, New York, NY. MR1662447

[4] Cressie, N. and G. Johannesson (2008). Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society Series B 70*, 209–226. MR2412639

[5] Dinur, I., G. Kindler, and R. R. Safra (2003). Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica 23*(2), 205–243. MR2001908

[6] Drineas, P., M. W. Mahoney, and S. Muthukrishnan (2006). Sampling algorithms for l_2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, 1127–1136. MR2373840

[7] Fithian, W. and T. Hastie (2014). Local case-control sampling: Efficient subsampling in imbalanced data sets. *The Annals of Statistics 42*(5), 1693–1724. MR3257627

[8] Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY. MR2722294

[9] He, X. (2017). Rotated sphere packing designs. *Journal of the American Statistical Association 112*(520), 1612–1622. MR3750885

[10] He, X. (2020). Lattice-based designs with quasi-optimal separation distance on all projections. *Biometrika 108*(2), 443–454. MR4259142

[11] Johnson, M. E., L. M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference 26*(2), 131–148. MR1079258

[12] Lin, Y. and H. H. Zhang (2006). Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics 34*(5), 2272–2297. MR2291500

[13] Ma, P., J. Z. Huang, and N. Zhang (2015). Efficient computation of smoothing splines via adaptive basis sampling. *Biometrika 102*(3), 631–645. MR3394280

[14] Mak, S. and V. Joseph (2018). Support points. *Annals of Statistics 47*(6A), 2562–2592. MR3851748

[15] Meng, C., X. Zhang, J. Zhang, W. Zhong, and P. Ma (2020). More efficient approximation of smoothing splines via space-filling basis selection. *Biometrika 107*(3), 723–735. MR4138986

[16] Picheny, V. and T. Wagner (2013). A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization 48*(3), 607–626.

[17] Rasmussen, C. E. and C. Williams (2005). *Gaussian Processes for Machine Learning*. MIT Press. MR2514435

[18] Tan, M. and Z. Zhang (2016). Wind turbine modeling with data-driven methods and radially uniform designs. *IEEE Transactions on Industrial Informatics 12*(3), 1261–1269.

[19] Wang, H., R. Zhu, and P. Ma (2018). Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association 113*(2), 1440037–1438957. MR3832230

[20] Wang, H. Y. (2019). Divide-and-conquer information-based optimal subdata selection algorithm. *Journal of Statistical Theory and Practice 13*(3), 46–64. MR3978445

[21] Wang, H. Y., M. Yang, and J. Stufken (2019). Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association 114*(525), 393–405. MR3941263

[22] Wang, L., J. Elmstedt, W. K. Wong, and H. Xu (2021). Orthogonal subsam-

pling for big data linear regression. *The Annals of Applied Statistics 15*(3), 1273–1290. MR4316648

[23] Wang, W., R. Tuo, and C. F. J. Wu (2020). On prediction properties of kriging: Uniform error bounds and robustness. *J. Am. Statist. Assoc 115*(530), 920–930. MR4107689

[24] Wu., S., X. Zhu., and H. Wang. (2021). Subsampling and jackknifing: A practically convenient solution for large data analysis with limited computational resources. *Statistica Sinica 10.5705/ss.202021.0257*. MR4607199

[25] Yao, Y. and H. Y. Wang (2019). Optimal subsampling for softmax regression. *Statistical Papers 60*(2), 585–599. MR3969047

[26] Yi, S. Y. and Y. D. Zhou (2023). Model-free global likelihood subsampling for massive data. *Statistics & Computing 33*(9). doi:10.1007/s11222-022-10185-0.