Statistics and Computing (2023) 33:9
https://doi.org/10.1007/s11222-022-10185-0

ORIGINALPAPER

Model-free global likelihood subsampling for massive data

Si-Yu Yi[1] · Yong-Dao Zhou[1]

Received: 27 August 2022 / Accepted: 20 November 2022 / Published online: 1 December 2022

Abstract
Most existing studies for subsampling heavily depend on a specified model. If the assumed model is not correct, the performance of the subsample may be poor. This paper focuses on a model-free subsampling method, called global likelihood subsampling, such that the subsample is robust to different model choices. It leverages the idea of the global likelihood sampler, which is an effective and robust sampling method from a given continuous distribution. Furthermore, we accelerate the algorithm for large-scale datasets and extend it to deal with high-dimensional data with relatively low computational complexity. Simulations and real data studies are conducted to apply the proposed method to regression and classification problems. It illustrates that this method is robust against different modeling methods and has promising performance compared with some existing model-free subsampling methods for data compression.

Keywords Data compression · Global likelihood sampler · Good lattice points · Space-filling design

1 Introduction

With the development of technology, massive data is more and more ubiquitous in many science fields. The storage memory and computing resource may be two challenging problems in conducting statistical analysis for massive data. Faced with those challenges, the subsampling techniques are studied broadly to extract useful information and attendantly reduce the burden of memory and computation (e.g., Mahoney 2011; Ma et al. 2014; Wang et al. 2018; Meng et al. 2021).

To improve the estimation efficiency, nonuniform subsampling methods are often used so that the data points with more information can be selected with higher probabilities. However, most existing studies focus on model-based subsampling methods which significantly depend on model assumptions. Ting and Brochu (2018) investigated optimal subsampling with influence functions. Wang et al. (2019) proposed the information-based optimal subdata selection (IBOSS) for linear models which selects the subsample deterministically. Wang and Ma (2021) studied optimal subsampling procedure that minimizes the asymptotic variance of the subsampling estimator for quantile regression. Yu et al. (2022) derived the MV and MVc methods which use the optimal Poisson subsampling probabilities in the context of quasi-likelihood estimation under the A- and L-optimality criteria. Wang et al. (2021) proposed an orthogonal subsampling approach for big data with a focus on linear regression models. When the model is correct, such methods can give an excellent compression of big data; otherwise, poor subsampling performance may result. In practical problems, a more common situation is that the underlying model is unknown before analyzing the data. Hence, with limited computational resource, a robust subsampling method against model misspecification may be a desirable choice.

A model-free subsampling method expects that the subsample has a statistical similarity to the full dataset. Joseph and Vakayil (2022) proposed an optimal data splitting method called SPlit for model validation. It splits the full dataset into training and testing sets such that the full dataset and the testing set follow the same distribution. Then, SPlit is also a model-free subsampling method. While, the computational cost of SPlit is fairly high especially when the data size is large or the dimension is high. To decrease the complexity, Vakayil and Joseph (2022) further studied a much quicker algorithm, called Twinning, to make it applicable to big data problems such as data compression. Vakayil and Joseph (2022) also theoretically showed that the objectives

✉ Yong-Dao Zhou
ydzhou@nankai.edu.cn

Si-Yu Yi
siyuyi@mail.nankai.edu.cn

[1] School of Statistics and Data Science, Nankai University, Tianjin 300071, China

of Twinning and SPlit are equivalent. Nevertheless, the performance of both SPlit and Twinning may become bad for data compression when the dimension is high. The complexity of Twinning would also be relatively high when the data size is large enough. In addition, Zhang et al. (2022) proposed a data-driven subsampling (DDS) method, which utilizes the rotation-inversion transformation based on a uniform design and takes the nearest neighbor substitution to obtain the subsample. Uniform design is a type of space-filling design under a given uniformity criterion, such as star discrepancy or mixture discrepancy. It aims to use a discrete point set to approximate the uniform distribution as closely as possible, see Fang et al. (2018) for details. DDS can generate subsample quickly and is a deterministic subsampling method, which requires the full data to have jointly independent coordinates.

The global likelihood sampler (GLS, Wang et al. 2015; Yi et al. 2022) is an effective and robust sampling method from a given distribution and easy to operate. It uses randomly-shifted good lattice points (GLP, Fang et al. 2018), a type of low-discrepancy point set, to generate several batches of the samples based upon the scaled densities on the design points. Multiple shifts of the GLP make the exploration of the target distribution more sufficient and hence GLS can well avoid falling into a local mode with cheap computation. Inspired by GLS, we propose a model-free subsampling strategy, called global likelihood subsampling (GLSS), such that the subsample can statistically mimic the full data as well as possible. GLSS is a deterministic subsampling method. Unlike GLS, we usually don't know the exact distribution of the data, and we need to estimate the data distribution first and then perform subsampling. In GLSS, a better performance can be obtained without multiple random shifts. The setting for the parameter in GLSS is more restrictive. Like GLS, a low-discrepancy point set can be used as the representative points in GLSS and we sample from these representative points with weights based on the estimated data distribution. Then the sequential nearest neighbor substitution is also applied to select a subset of the original full data. Moreover, GLSS relaxes the limitation on the independence of each dimension of the data in DDS. Furthermore, we propose some variants of GLSS to accelerate the algorithm for large-scale datasets and deal with the high-dimensional data with low computational complexity. In terms of the prediction performance under different models, compared with the uniform random sampling (URS), a common baseline for developing model-free methods, GLSS has a significant performance improvement. The computational complexity of GLSS is much lower than SPlit and GLSS has comparable performance with SPlit in many low-dimensional cases. With a well-set parameter, GLSS outperforms Twinning in low-dimensional cases. In some high-dimensional cases, GLSS can perform better than both SPlit and Twinning. It can also perform much better than DDS in many cases. Moreover, comparing GLSS with some model-based subsampling methods such as IBOSS, MV, and MVc, GLSS shows a better effect and robustness when the evaluation criterion is inconsistent with the criterion which derives the subsampling rule. Further, GLSS may have potential applications in other domains, such as the scalable Markov chain Monte Carlo (Maire et al. 2019), in which subsampling is essential to reduce the computational budget.

This paper is organized as follows. Section 2 describes the GLS algorithm and proposes the GLSS algorithm. Some details for GLSS and the variants of GLSS are discussed. The convergence result of GLSS is derived in Sect. 3. To compare the model-robust property of URS, SPlit, Twinning, DDS, IBOSS, MV, MVc, and GLSS for data compression under different models, Sects. 4 and 5 show some simulation examples and real data studies for regression and classification. Finally, some conclusions and discussions are given in Sect. 6. All the proofs of the lemmas and theorems are given in the Appendix.

2 Methodology

In this section, we briefly introduce the GLS algorithm and propose the GLSS algorithm for subsampling. A simple acceleration strategy and a dimension reduction technique are embedded into GLSS for large-scale datasets and high-dimensional data with low complexity, respectively.

2.1 Global likelihood sampler

Let $F(\boldsymbol{\theta})$ be a d-dimensional cumulative distribution function (CDF) and $f(\boldsymbol{\theta})$ be the corresponding probability density function (PDF). Following Yi et al. (2022), GLS first truncates a closed and bounded hypercube $\mathcal{D}_0$ with most of the probabilities of interest contained and an M-point GLP is randomly shifted $\tilde{n}$ times in the truncated hypercube. In the ith shift for $i = 1, \ldots, \tilde{n}$, GLS defines the weights at the randomly-shifted GLP $\{\boldsymbol{g}_k^{(i)} \in \mathcal{D}_0 : k = 1, \ldots, M\}$ based on the scaled likelihoods $f(\boldsymbol{g}_k^{(i)})/\sum_{j=1}^{M} f(\boldsymbol{g}_j^{(i)})$ for $k = 1, \ldots, M$ and then it performs multinomial sampling associated with the weighted shifted-GLP to generate an $\tilde{m}$-size sample. Based on $\tilde{n}$ randomly shifts on the GLP, there are $\tilde{n}$ multinomial samplings to generate $\tilde{n}$ batches of $\tilde{m}$-size samples. After running GLS, an $\tilde{n}\tilde{m}$-size GLS sample will be produced.

In GLS, by using the GLP, there is no need for the proposal distribution and thus it avoids the problems caused by poor proposal distribution. It makes that GLS can be problem-independent for different target distributions. Compared with the case where GLP is replaced by the uniformly

distributed random proposals, GLS can produce more representative weighted points for the target distribution since the (randomly-shifted) GLP has better space-filling property. Multiple random shifts and multinomial samplings increase the diversity and randomness of the sampling. Hence, GLS can be effective to explore the structure of complicated target distribution. Yi et al. (2022) also theoretically proved that the empirical cumulative distribution function (ECDF) of the GLS sample uniformly converges to the target distribution in probability and with probability 1 under some conditions. When the one-dimensional random numbers on $[0, 1]$ in the multinomial sampling of each batch are replaced by the randomized low-discrepancy point set $\{((2j-1)/(2\tilde{m}) + u) \bmod 1 : u \sim U(0,1), j = 1, \ldots, \tilde{m}\}$, the rate for convergence in probability can be improved. By simulations and applications in Yi et al. (2022), the effectiveness, robustness, and speediness of GLS are demonstrated by comparisons with some other commonly used sampling methods, e.g., sampling/importance resampling (Rubin 1987), Metropolis-Hastings algorithm (Hastings 1970) and evolutionary Monte Carlo (Liang and Wong 2001).

2.2 Global likelihood subsampling

In this paper, under the model-free assumption, our purpose is to extract an n-size subsample $\tilde{\mathcal{Z}}$ from the N-size full data $\mathcal{Z}$, such that $\tilde{\mathcal{Z}}$ can statistically represent $\mathcal{Z}$ for data compression. Suppose that $\mathcal{Z}$ contains N independent and identically distributed $(i.i.d.)$ samples coming from F over $\mathbb{R}^d$. According to the GLS algorithm, we need to know the target distribution up to a multiplicative constant for sampling. In the subsampling issue, we may not achieve the CDF F directly. Instead, we adopt the kernel density estimation (KDE, Davis et al. 2011; Parzen 1962) to estimate the density function for $\mathcal{Z}$. Then, we perform subsampling based on the estimated density function. For any $\boldsymbol{\theta} \in \mathbb{R}^d$, with a specific kernel function, the KDE based on $\mathcal{Z}$ is given by

$$\hat{f}_{\mathrm{K}}(\boldsymbol{\theta}) = \frac{1}{N \cdot r^d} \sum_{i=1}^{N} K\left(\frac{\boldsymbol{\theta} - \boldsymbol{z}_i}{r}\right),$$
$$K(\boldsymbol{u}) = \begin{cases} 1/v_d & ||\boldsymbol{u}|| \le 1, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

where $\boldsymbol{z}_i \in \mathcal{Z}$ for $i = 1, \ldots, N$, r is the bandwidth, K is the uniform kernel, and v_d is the volume of a unit ball in $\mathbb{R}^d$. Hence, the estimated density at $\boldsymbol{\theta}$ is proportional to the number of data points in the ball $\mathcal{B}(\boldsymbol{\theta}, r)$ with $\boldsymbol{\theta}$ as the center and r as the radius.

Based on $\hat{f}_{\mathrm{K}}$, we wish to sample from the representative points for the data space $\mathcal{D}$ (i.e., a bounded hypercube such that $\mathcal{Z} \subseteq \mathcal{D}$), which can fill the space well and spread as uniformly as possible. Inspired by GLS, we utilize the space-filling design $\mathcal{S} = \{\boldsymbol{s}_i : i = 1, \ldots, M\}$ in $\mathcal{D}$, e.g., the low-discrepancy point set GLP, and estimate the densities of the design points by $\hat{f}_{\mathrm{K}}$, in which r is chosen as half of the separation distance (i.e., the minimal pairwise distance) of $\mathcal{S}$. For any $1 \le k \le M$, $(\boldsymbol{s}_k, \hat{f}_{\mathrm{K}}(\boldsymbol{s}_k))$ can be regarded as a representation for the data information in $\mathcal{B}(\boldsymbol{s}_k, r)$. Then, similar to the improved GLS in Yi et al. (2022), to yield more representative samples, instead of the uniform random numbers, we can use the one-dimensional low-discrepancy point set $\{(2j-1)/2n : j = 1, \ldots, n\}$ to do multinomial sampling on the distribution placing mass $\hat{f}_{\mathrm{K}}(\boldsymbol{s}_k)/\sum_{i=1}^{M} \hat{f}_{\mathrm{K}}(\boldsymbol{s}_i)$ at $\boldsymbol{s}_k, k = 1, \ldots, M$, where n is the subsample size. The more data points around the design point, the higher the probability that the design point would be drawn. Empirically, compared with GLS, the use of the space-filling design without random shifts can attain a better performance for subsampling. It can not only enable $\tilde{\mathcal{Z}}$ to capture the main data information in $\mathcal{Z}$, but also preserve the pattern of $\mathcal{S}$ to a certain extent. Furthermore, for the subsampling purpose, the final subsample should be a subset of the full data. To obtain the final subsample, for any drawn design point $\boldsymbol{d}_i, i = 1 \ldots, n$, we use the sequential nearest neighbor substitution in the small region $\mathcal{B}(\boldsymbol{d}_i, r)$. Specifically, we first record and count distinct elements in $\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n\}$ by $[\{\bar{\boldsymbol{d}}_1, \ldots, \bar{\boldsymbol{d}}_{n_\mathrm{u}}\}, \{c_1, \ldots, c_{n_\mathrm{u}}\}] = \texttt{unique\_count}(\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n\})$, where $\bar{\boldsymbol{d}}_q$ is the qth distinct point and c_q is the corresponding number of times that $\bar{\boldsymbol{d}}_q$ appears in $\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n\}$ for $q = 1, \ldots, n_\mathrm{u}$. Then, for any $\bar{\boldsymbol{d}}_q, q = 1, \ldots, n_\mathrm{u}$, we record the cardinality of $\{\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}\}$ by $C_q = \#\{\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}\}$; we compute the quotient and remainder of c_q/C_q by $[\mathrm{quo}_q, \mathrm{rem}_q] = \texttt{division}(c_q, C_q)$; we repeat the data points in $\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}$ quo_q time(s) by $\texttt{repeat}(\mathrm{quo}_q, \{\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}\})$ and find the rem_q nearest data points to $\bar{\boldsymbol{d}}_q$ in $\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}$ to construct partial subsample based on $\bar{\boldsymbol{d}}_q$. With a moderate value for M, if the full data is not extremely imbalanced, quo_q is usually not larger than or equal to 1.

It allows the final subsample to be as non-repetitive as possible. Since the L_2-norm between each drawn design point and the corresponding replacing data point is no more than r, the final subsample can statistically mimic the distribution of drawn design points to some extent.

To intuitively show the whole procedure of GLSS, we give the pseudo-code in Algorithm 1. GLSS is a deterministic subsampling method. In contrast to GLS, the requirement on the setting of M, the run size of the space-filling design $\mathcal{S}$, is more restrictive in GLSS. According to the results in Sects. 4 and 5, M mainly depends on the number of dimensions d, the full data size N and the subsample size n. We may approximately set $M = \exp(2(d+\nu))$ with $-d < \nu \le 2$. When d is larger, ν is smaller for controlling the computational complexity. Under the same d, M becomes larger with the increase of N or n for capturing the data structure better. In the imple-

Algorithm 1 The pseudo-code of GLSS

Input: Full data, $\mathcal{Z} = \{z_i \in \mathcal{D} : i = 1, \ldots, N\}$; Space-filling design, $\mathcal{S} = \{\boldsymbol{s}_i \in \mathcal{D} : i = 1, \ldots, M\}$; the separation distance of $\mathcal{S}$, $r_{\mathcal{S}}$; the size of the final subsample, n.
Output: Subsample $\tilde{\mathcal{Z}} = \{\tilde{z}_i \in \mathcal{D} : i = 1, \ldots, n\}$;
Initialize: $\tilde{\mathcal{Z}} = \emptyset$ and $r = r_{\mathcal{S}}/2$;
for $j = 1$ to n **do**
  $l = (2j-1)/(2n)$;
  for $k = 1$ to M **do**
    if $\sum_{i=1}^{k-1} \hat{f}_{\mathrm{K}}(\boldsymbol{s}_i) < l \sum_{i=1}^{M} \hat{f}_{\mathrm{K}}(\boldsymbol{s}_i) \le \sum_{i=1}^{k} \hat{f}_{\mathrm{K}}(\boldsymbol{s}_i)$ **then**
      $\boldsymbol{d}_j = \boldsymbol{s}_k$;
      break
    end if
  end for
end for
$[\{\bar{\boldsymbol{d}}_1, \ldots, \bar{\boldsymbol{d}}_{n_{\mathrm{u}}}\}, \{c_1, \ldots, c_{n_{\mathrm{u}}}\}] = \mathrm{unique\_count}(\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n\})$;
for $q = 1$ to n_{u} **do**
  $C_q = \#\{\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}\}$; $[\mathrm{quo}_q, \mathrm{rem}_q] = \mathrm{division}(c_q, C_q)$;
  $\tilde{\mathcal{Z}} = \tilde{\mathcal{Z}} \cup \mathrm{repeat}(\mathrm{quo}_q, \{\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}\}) \cup \mathrm{argmin}_{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\mathrm{rem}_q} \in \mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}} \sum_{i=1}^{\mathrm{rem}_q} ||\boldsymbol{v}_i - \bar{\boldsymbol{d}}_q||\}$;
end for
return $\tilde{\mathcal{Z}}$;

mentation, if $\mathcal{S}$ is chosen as a GLP, we can use the *Lattice* function (Nuyens and Cools 2006) in the *mined* R package (Wang and Joseph 2022) to fast construct it, in which the size M should be set as a prime integer. $\mathcal{S}$ can be substituted by other space-filling designs under the discrepancy-based or distance-based criteria (Fang et al. 2018). Hereafter, we choose a M-size GLP as $\mathcal{S}$ to perform GLSS. On the other hand, based on KDE and the chosen space-filling design, we need to calculate the density at each design point quickly for the subsequent sampling, especially when M is large. To meet this requirement, the k-d tree (Bentley 1975) is a proper choice, which is a multi-dimensional binary search tree as a data structure for the storage of information to be retrieved by associative searches. This structure is efficient for several applications, such as range searches and nearest neighbor searches. Hence, in GLSS, we can use the k-d tree to calculate $r_{\mathcal{S}}$, $\hat{f}_{\mathrm{K}}$, and the rem_q nearest data points for $\bar{\boldsymbol{d}}_q, q = 1, \ldots, n_{\mathrm{u}}$ to accelerate the algorithm. In addition, the parallel computation technique can be used in each FOR loop of GLSS to further speed up the algorithm.

Based on the use of the k-d tree, we analyze the computational complexity of GLSS as follows. The complexity of building a k-d tree from N d-dimensional points is $O(dN \log N)$. We use the fixed-radius near neighbors query to calculate the estimated density $\hat{f}_{\mathrm{K}}$ at the M points in $\mathcal{S}$ with average-case complexity $O(\sum_{i=1}^{M}(\log N + k_{\mathrm{r},i}))$ and worst-case complexity $O(\sum_{i=1}^{M}(N^{1-1/d} + k_{\mathrm{r},i}))$, where $k_{\mathrm{r},i}$ is the number of reported points in $\mathcal{B}(\boldsymbol{s}_i, r)$ for $i = 1, \ldots, M$. Both the average-case and worst-case complexity of each multinomial sampling are $O(M)$. We can record and count distinct elements in $O(dn)$ time on average and $O(dn \log n)$ time in the worst case. Moreover, we need to build the k-d tree in $\mathcal{B}(\bar{\boldsymbol{d}}_q, r) \cap \mathcal{Z}, q = 1, \ldots, n_{\mathrm{u}}$ for the rem_q-nearest neighbors query, and search the neighbors with average-case complexity $O(dC_q \log C_q + \log C_q + \mathrm{rem}_q)$ and worst-case complexity $O(dC_q \log C_q + C_q^{1-1/d} + \mathrm{rem}_q)$. Note that $n < N$, $\sum_{i=1}^{M} k_{\mathrm{r},i} \le N$, $\sum_{q=1}^{n_{\mathrm{u}}} C_q \le N$ and $\sum_{q=1}^{n_{\mathrm{u}}} \mathrm{rem}_q \le N$. Thus, the overall time complexity of GLSS in the average case can be simplified to $O(dN \log N) + O(M \log N + \sum_{i=1}^{M} k_{\mathrm{r},i}) + O(M) + O(dn) + O(\sum_{q=1}^{n_{\mathrm{u}}}(dC_q \log C_q + \log C_q + \mathrm{rem}_q)) = O(dN \log N + M \log N)$, while the worst-case complexity of GLSS is $O(dN \log N) + O(MN^{1-1/d} + \sum_{i=1}^{M} k_{\mathrm{r},i}) + O(M) + O(dn \log n) + O(\sum_{q=1}^{n_{\mathrm{u}}}(dC_q \log C_q + C_q^{1-1/d} + \mathrm{rem}_q)) = O(dN \log N + MN^{1-1/d})$. In the following, we consider two 2-dimensional toy examples for visualization.

Example 1 Consider two different datasets, as shown in Fig. 1, i.e., (i) the full data $\mathcal{Z}$ is generated from a two-dimensional standard normal distribution; (ii) each element (x_i, y_i) in the full data follows $y_i = x_i^2 + \epsilon_i, x_i \sim \mathcal{N}(0, 1), \epsilon_i \sim \mathcal{N}(0, 1)$ for $i = 1, \ldots, N$. Let the sizes of the full data and the final subsample be $N = 2000$ and $n = 60$, respectively. Based on the two datasets, we compare URS with GLSS. In GLSS, the sizes of the used GLPs in the two cases are 349 and 887, respectively. The data structure in case (i) is simple and n is small, so a GLP with a small size is enough for GLSS, while for the data with a more complicated structure, e.g., case (ii), GLSS needs a larger size for GLP to achieve a good performance. From Fig. 1, it can be seen that for the two datasets, the subsample under GLSS gives a better representation of the full data compared with URS.

In application, if N is not very large, e.g., N <5e5= 5×10^5, or the corresponding computational complexity is acceptable, we can directly perform GLSS on $\mathcal{Z}$ for good subsampling performance. Otherwise, to reduce the computational cost, a compromising strategy is to first conduct URS on $\mathcal{Z}$ to obtain an N_{s}-size ($N > N_{\mathrm{s}} > n$) intermediate subsample $\mathcal{Z}_{\mathrm{s}}$, and then perform GLSS on $\mathcal{Z}_{\mathrm{s}}$ to achieve the final subsample $\tilde{\mathcal{Z}}$, i.e., run Algorithm 1 with $\mathcal{Z}$ being replaced by $\mathcal{Z}_{\mathrm{s}}$. We denote this procedure by $\mathrm{GLSS_a}$. In this case, the setting of M in $\mathrm{GLSS_a}$ shall be closely associated with d, N_{s}, and n. The computational complexity of $\mathrm{GLSS_a}$ should be $O(dN_{\mathrm{s}} \log N_{\mathrm{s}} + M \log N_{\mathrm{s}})$ on average and $O(dN_{\mathrm{s}} \log N_{\mathrm{s}} + MN_{\mathrm{s}}^{1-1/d})$ in the worst case. The set of N_{s} shall be a tradeoff between the subsampling performance and time complexity. By the empirical results in Sect. 4.1, when $N >$ 5e5, $N_{\mathrm{s}} = N/10$ is an appropriate choice for good performance and reduced running cost.

In addition, we may not know the specific structure of the full data in applications. To better catch the data information, with the increase of d, the size M of $\mathcal{S}$ in GLSS would increase exponentially. The attendant challenge is the high computational complexity due to large d and M. To handle this problem, dimension reduction techniques should be

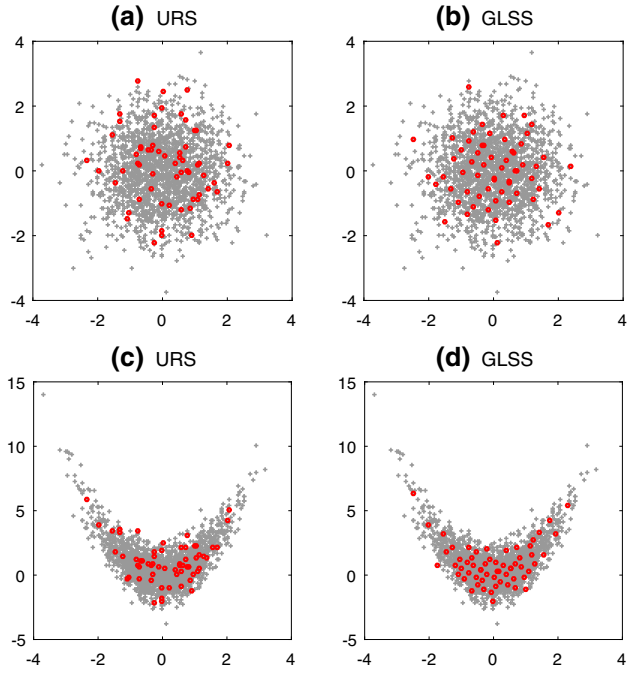

Fig. 1 The left and right panels are the results under URS and GLSS, respectively. '+': the original synthetic data; 'o': the subsample

taken into account and we propose a high-dimensional version of GLSS based on singular value decomposition (SVD) and denote it as GLSS_h. We do not directly implement GLSS on the high-dimensional full data. Instead, we perform SVD on the normalized full data for reducing the dimension of the data and conduct GLSS in the reduced latent space. Then, we convert the subsample back to the original space. Specifically, GLSS_h can be described in the following three steps.

Step 1 Normalize the full data $\mathcal{Z}$ to obtain $\tilde{\mathcal{Z}}$ and perform SVD on $\tilde{\mathcal{Z}}$ as $\tilde{\mathcal{Z}}_{N\times d} = U_{N\times d}S_{d\times d}V_{d\times d}^\top$. Denote by U' the first $d'(< d)$ columns of U;

Step 2 Implement the GLSS algorithm on the latent space U' and record the indexes of the selected rows with size n;

Step 3 Select the rows in the full data $\mathcal{Z}$ corresponding to the indexes in Step 2 to obtain the final subsample.

In practical application, we recommend using GLSS_h when $d \geq 5$ and choosing the minimal number of the dimensions that can make the contribution accounting for the total variation of the full data reach 70%, as the d'. By the empirical results in Sect. 4, the setting of M in Step 2 may be relative to both d' and d for a good performance, that is, we can set $M = \exp(2(d' + \nu))$ with $-d' < \nu \leq 2$. According to the simulation study, it is recommended that $\nu = 1.4, 1.2, 0.6, 0.3$ for $d' = 2, 3, 4, 5$, respectively. For larger d', we limit $5.3-d' < \nu < 6-d'$ to balance the performance and complexity. With the same d', M becomes larger when d is larger. Performing SVD requires $O(d^2N)$ time. Hence, in this case, the computational complexity of GLSS_h should be $O(d^2N + d'N\log N + M\log N)$ in the average case and $O(d^2N + d'N\log N + MN^{1-1/d'})$ in the worst case. For ultrahigh-dimensional data, the variable selection techniques, e.g., LASSO (Tibshirani 1996) or Pearson correlation coefficient comparison (Fan and Lv 2008), can also be incorporated before performing SVD, and then we only focus on the important variables in SVD, subsampling, and modeling. In general, the number of the significant variables would not be large and the requirement for the contribution rate can be approximately met with relatively small d' in real data. The simulations and real data studies in Sects. 4 and 5 illustrate that with a moderate number of important variables, $d' = 2$ is a reasonable and suitable choice in many cases with good performance and low complexity for data.

Synthetically, when $N > 5\text{e}5$ and $d \geq 5$, we can combine GLSS_a with GLSS_h and denote it as GLSS_ah, i.e., we first perform URS on the N-size full data $\mathcal{Z}$ to obtain the N_s-size subsample $\mathcal{Z}_\text{s}$ before Step 1 of GLSS_h and replace the full data in Steps 1 and 3 with the N_s-size subsample $\mathcal{Z}_\text{s}$. In this case, the setting of M in GLSS_ah should be mainly relative to d', N_s and n. The time complexity of GLSS_ah is $O(d^2N_\text{s} + d'N_\text{s}\log N_\text{s} + M\log N_\text{s})$ and $O(d^2N_\text{s}+d'N_\text{s}\log N_\text{s}+MN_\text{s}^{1-1/d'})$ in the average and worst cases, respectively.

3 Theoretical results

In this section, we establish the convergence result for GLSS based on the theoretical results for KDE and GLS. First, for the estimated density by KDE, based on (1) and Jiang (2017), we can obtain the following result.

Lemma 1 *If f is bounded and Hölder-continuous (i.e., $|f(\boldsymbol{\theta}) - f(\boldsymbol{\theta}')| \leq C_\alpha||\boldsymbol{\theta} - \boldsymbol{\theta}'||^\alpha$ for $\boldsymbol{\theta}, \boldsymbol{\theta}'$ in $\mathbb{R}^d$ and $0 < \alpha \leq 1$), and the support of f is bounded, then uniformly for $r > (\log N/N)^{1/d}$, we have*

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |\hat{F}_\text{K}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| = O_\text{P}(\Delta) \ \textit{with} \ \Delta = r^\alpha + \sqrt{\frac{\log N}{Nr^d}},$$

where $\hat{F}_\text{K}$ is the corresponding CDF of $\hat{f}_\text{K}$.

Lemma 1 is a direct application of Theorem 2 in Jiang (2017). It theoretically guarantees the effectiveness of the multivariate KDE. Next, under the multinomial sampling based on $\mathcal{S}$ and the corresponding weights defined by KDE, we denote by $\tilde{\mathcal{Z}}_\text{s}$ the n-point drawn design points. Let $F_{\tilde{\mathcal{Z}}_\text{s}}$ be the ECDF of $\tilde{\mathcal{Z}}_\text{s}$ and ϱ be the mapping for the multinomial sampling. Denote $\varrho_{\boldsymbol{\theta}} := I_{[-\infty,\boldsymbol{\theta}]} \circ \varrho$, where $\circ$ represents the

composition of two functions. Based on $\hat{f}_{\mathrm{K}}(\boldsymbol{\theta})$ in (1), the sampling process in GLSS is similar to that in GLS, except that $\mathcal{S}$ is not randomly shifted multiple times in GLSS. Hence, following Yi et al. (2022), we can obtain the uniform convergence result for $F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}$ as follows.

Lemma 2 *If the following conditions are satisfied,*

(1) $\hat{f}_{\mathrm{K}}(\boldsymbol{x})I_{[-\infty,\boldsymbol{\theta}]}(\boldsymbol{x})$ *is of uniformly bounded variation in the sense of Hardy and Krause for any* $\boldsymbol{\theta} \in \mathbb{R}^d$,
(2) $\mathcal{S}$ *is a low-discrepancy point set,*
(3) $\varrho_{\boldsymbol{\theta}}(u)$ *is of uniformly bounded variation in the sense of Hardy and Krause for any* $\boldsymbol{\theta} \in \mathbb{R}^d$ *and run size* M *of* $\mathcal{S}$,

then we have

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - \hat{F}_{\mathrm{K}}(\boldsymbol{\theta})| = O\left(\max\left\{\kappa(M), \frac{1}{n}\right\}\right),$$

where $\kappa(M) = 1/M$ *if* $d = 1$, *and* $(\log(M))^d/M$, *otherwise.*

The proof of Lemma 2 follows Lemma 1(1) and Theorem 2 in Yi et al. (2022). In essential, the order O(1/n) is consistent with the result of error bound in the quasi Monte Carlo method by the famous Koksma-Hlawka inequality (Hlawka 1961). By Lemma 2, it theoretically implies that the ECDF of the drawn design points converges to the estimated CDF for $\mathcal{Z}$ based on KDE. Furthermore, combining Lemma 1 with Lemma 2, we can obtain the convergence between $F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}$ and F as follows.

Theorem 3 *If the conditions in Lemmas 1 and 2 are satisfied, we have*

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| = O_{\mathrm{P}}(\Lambda),$$

where $\Lambda = \max\{1/M^{\alpha/d} + (M\log N/N)^{1/2}, 1/n\}$.

Remark 1 Suppose that the conditions in Theorem 3 are satisfied, (i) if $M = O((N/\log N)^{d/(d+2\alpha)})$, we have $\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| = O_{\mathrm{P}}(\max\{1/M^{\alpha/d}, 1/n\})$; (ii) if $M = O((N/\log N)^{d/(d+2\alpha)})$ and $n = O(M^{\alpha/d})$, we further have $\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| = O_{\mathrm{P}}(1/M^{\alpha/d})$.

From Theorem 3, the ECDF of the drawn design points can uniformly converge to the true CDF of the full data. Remark 1 is a direct application of Theorem 3, in which the convergence rate can be simplified when imposing some conditions on M and n. Then, by the sequential nearest neighbor substitution as described in Algorithm 1, the final subsample can also be statistically representative of the true distribution for the full data and thus of the full data, if the disturbance error can be ignored.

Finally, for the variant $\mathrm{GLSS_a}$, let $F_{\mathcal{Z}_{\mathrm{s}}}$ be the ECDF of the N_{s}-size intermediate subsample $\mathcal{Z}_{\mathrm{s}}$. We show that the URS strategy in $\mathrm{GLSS_a}$ is distribution-preserving in the following.

Lemma 4 *If the elements in* $\mathcal{Z}$ *follow* F *and we perform URS with replacement on* $\mathcal{Z}$ *to obtain the* N_{s}*-size* $\mathcal{Z}_{\mathrm{s}}$, *then the elements in* $\mathcal{Z}_{\mathrm{s}}$ *also follow* F *and* $\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F(\boldsymbol{\theta}) - F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta})| = O_{\mathrm{P}}(1/\sqrt{N_{\mathrm{s}}})$.

Thus, based on Lemma 4, the conclusions in Lemmas 1 and 2, Theorem 3, and Remark 1 also hold if $\{\mathcal{Z}, N\}$ is replaced by $\{\mathcal{Z}_{\mathrm{s}}, N_{\mathrm{s}}\}$ since the $i.i.d.$ samples in $\mathcal{Z}_{\mathrm{s}}$ also follow F. It implies that $\mathrm{GLSS_a}$ should be effective for large N and not too small N_{s}.

4 Numerical examples

As mentioned earlier, by GLSS, we want to find a subsample $\tilde{\mathcal{Z}}$ from the full data $\mathcal{Z}$ such that $\tilde{\mathcal{Z}}$ is a good representation of $\mathcal{Z}$ for reducing the storage memory and decreasing the computational complexity of modeling. In this section, we perform GLSS under different settings and compare it with some other model-free or model-based subsampling methods to illustrate its validity for data compression.

In Sects. 4.1 and 4.2, we compare the performance and running time of GLSS, $\mathrm{GLSS_a}$, and $\mathrm{GLSS_h}$ with those of some other model-free methods, i.e., URS, SPlit, Twinning, and DDS, under different subsample sizes, data sizes, numbers of dimensions, and models. We also make the comparison between $\mathrm{GLSS_h}$ with some model-based methods, i.e., IBOSS, MV, and MVc, in Sect. 4.3. Sections 4.4 and 4.5 study the effects of the contribution rate by SVD and the effective number of dimensions in $\mathrm{GLSS_h}$.

We use the *SPlit* R package (Vakayil et al. 2022) to perform SPlit in parallel, in which the maximum number of iterations is 500 and the tolerance level is 1e-6. The *twinning* R package (Vakayil et al. 2022) is used for conducting Twinning in parallel. Other settings for SPlit and Twinning are set as the default values in the R packages. In DDS, we adopt the n-size GLP as the uniform template. In GLSS, the value of M is set to a prime integer for ease of constructing the corresponding GLP by the *Lattice* function, as discussed in Sect. 2.2.

All the variables in the sets to be subsampled are normalized to mean 0 and standard deviation 1 before subsampling. We repeat each subsampling method 50 times. For DDS, GLSS, $\mathrm{GLSS_a}$, and $\mathrm{GLSS_h}$, we shift the used GLP by a small-valued uniformly distributed random vector in each repetition. Except for Sect. 4.3, all the subsampling procedures are performed on the N_{tr}-size training sets (i.e., $N = N_{\mathrm{tr}}$) from different data-generating models. Based on the obtained subsample under each method, to compare the model-free property, i.e., the robustness against different

modeling methods, we consider the linear regression model, the Gaussian process model (GP, Santner et al. 2003) and the multivariate adaptive regression splines (MARS, Friedman 1991). In each modeling method, the parameter settings for different subsamples from different subsampling methods are the same for a fair comparison. To measure the prediction performance of these fitted models, we compute the logarithm of the rooted mean squared prediction error (log RMSPE) based on the testing set, i.e., $\log \text{RMSPE} = \log((\sum_{i=1}^{N_{\text{te}}}(y_i - \hat{y}_i)^2/N_{\text{te}})^{1/2})$, where N_{te} is the size of the testing set, y_i is the ith response in the testing set, and $\hat{y}_i$ is the prediction of y_i based on the fitted model for $i = 1, \ldots, N_{\text{te}}$. The evaluation criterion is also used in Joseph and Vakayil (2022); Vakayil and Joseph (2022); Zhang et al. (2022). The time complexity is compared by recording the logarithm of the average CPU time for running each subsampling method once in parallel (log Time) in Sects. 4.1 and 4.2. In Sect. 4.3, we perform subsampling on the full data. Following Yu et al. (2022), based on the subsamples from GLSS_{h}, IBOSS, MV, and MVc, we compare the logarithm of the empirical MSE (log MSE) of the parameter estimator of interest (e.g., the regression coefficient and the population mean), i.e., $\log \text{MSE} = \log(\sum_{i=1}^{50} ||\hat{\boldsymbol{\xi}}_{(i)} - \hat{\boldsymbol{\xi}}_{\text{full}}||^2/50)$, where $\hat{\boldsymbol{\xi}}_{(i)}$ is the parameter estimator from the ith subsample under some subsampling method, and $\hat{\boldsymbol{\xi}}_{\text{full}}$ is the estimator from the full data. Computations are carried out on a normal PC with a 6-core 2.4 GHz Intel processor.

4.1 Comparison of GLSS, GLSS_a and URS, SPlit, twinning, DDS

We consider that the data is generated by a univariate second-order model

$$y(x) = x^2 + \epsilon, x \sim \mathcal{N}(0, 1), \quad \epsilon \sim \mathcal{N}(0, 1), \tag{2}$$

which corresponds to a 2-dimensional subsampling problem, as shown in Fig. 1. We first fix $N_{\text{tr}} = 1\text{e}4$, $N_{\text{te}} = 5\text{e}4$ and let $n = 300, 600, 900, 1200$. We generate both the training set and the testing set by the model (2). For comparison, we conduct URS, SPlit, Twinning, DDS, and GLSS on the training set. In GLSS, we set $M = 881, 1031, 1193, 1301$ for each n in turn. Based on the subsamples, the linear, GP and MARS models are considered, in which the linear model and the mean function in the GP model are fitted using the second-order term x^2. Figure 2 shows all the prediction results under different modeling methods using the n-size subsamples from different subsampling methods. Due to the dependence between x and y, DDS shows the worst performance, even if the PCA technique is integrated into the subsampling. Both SPlit and Twinning perform better than URS and DDS, whereas Twinning is worse than SPlit in many cases and due to the random choice of the initial value, it is less stable than SPlit with larger averages and variances of log RMSPEs. As for GLSS, it also outperforms URS and DDS with smaller log RMSPEs. Compared with SPlit, GLSS shows comparable and promising performance. Compared with Twinning, GLSS has better worst-case performance

for the three considered regression methods. In addition, it can be seen from Fig. 2 that whatever n is, the prediction result can lead to a similar conclusion by comparing the log RMSPEs among different subsampling methods based on each modeling method. It indicates that the subsample size n has little impact on the comparison of these subsampling methods.

Then, we fix $n = 300$, $N_{\text{te}} = 5\text{e}4$ and let $N_{\text{tr}} = 5\text{e}4, 1\text{e}5, 5\text{e}5, 1\text{e}6, 5\text{e}6$. We perform URS, SPlit, Twinning, DDS, GLSS, and GLSS_{a} on the training sets with different sizes. In GLSS, we set $M = 1307, 1307, 1307, 1361, 1609$ for $N_{\text{tr}} = 5\text{e}4, 1\text{e}5, 5\text{e}5, 1\text{e}6, 5\text{e}6$, respectively. In GLSS_{a}, we set $N_{\text{s}} = N_{\text{tr}}/10$ and $M = 853, 857, 1301, 1301, 1381$ for each N_{tr} in turn. Figure 3 shows the prediction result using each obtained subsample and the average running time of each subsampling method. It can be seen that the result for each N_{tr} gives similar conclusions by comparing the log RMSPEs of each fitted model based on URS, SPlit, Twinning, DDS, and GLSS. The SPlit brings relatively good prediction result, while it occupies the highest computational cost, especially when N_{tr} is large. Twinning is less stable and has a larger log RMSPE than SPlit. GLSS shows promising performance with a much lower time cost compared with SPlit and it also has better performance than Twinning. For GLSS_{a}, when $N_{\text{tr}} > 5\text{e}5$, the prediction abilities based on GLSS_{a} under the three modeling methods are comparable in contrast with GLSS, whereas with $N_{\text{tr}} < 5\text{e}5$, GLSS_{a} performs slightly worse, which implies that GLSS_{a} is more effective when N_{tr} is larger. In terms of the running time, it can be seen from Fig. 3f that GLSS would be quicker than Twinning when $N_{\text{tr}} > 1\text{e}6$, while Twinning is faster if $N_{\text{tr}} < 1\text{e}6$. The running costs of DDS and Twinning are similar when $N_{\text{tr}} < 1\text{e}6$, whereas Twinning spends a higher cost when N_{tr} becomes larger. Moreover, GLSS_{a} can sharply reduce the running time for large N_{tr} compared with GLSS.

4.2 Comparison of GLSS_{h} and URS, SPlit, twinning, DDS

We consider a multivariate data-generating model as

$$y(\boldsymbol{x}) = (x_1 + x_2 + \cdots + x_{d-1})^2 + \epsilon,$$
$$x_1, x_2 \overset{i.i.d}{\sim} \mathcal{N}(0, 1), \epsilon \sim \mathcal{N}(0, 1), \tag{3}$$

where x_i is generated by $x_i = x_{i-1} + x_{i-2}$ for $i = 3, \ldots, d-1$. In the model (3), only x_1 and x_2 are independent and the generated data corresponds to a d-dimensional subsampling

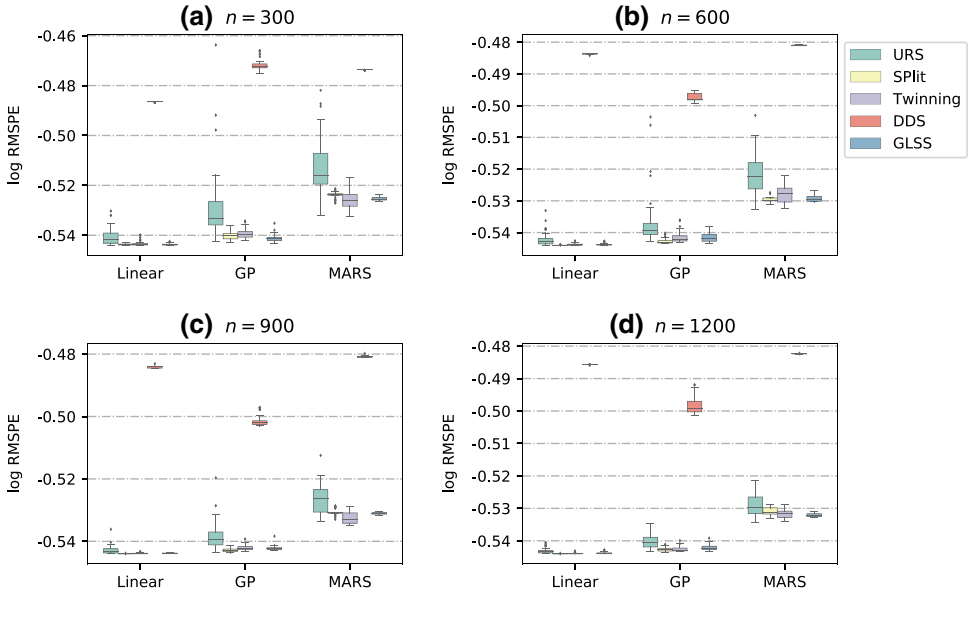

Fig. 2 The comparison of the log RMSPEs for different subsample sizes and models under URS, SPlit, Twinning, DDS, and GLSS in Sect. 4.1

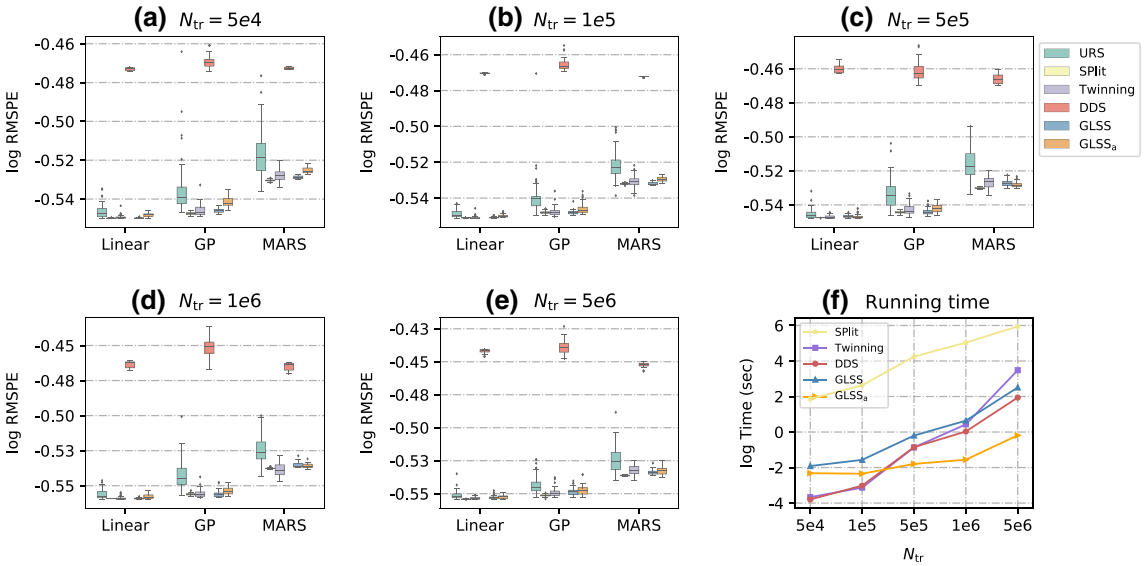

Fig. 3 The comparison of **a**–**e** the log RMSPEs for different data sizes and models under URS, SPlit, Twinning, DDS, GLSS, and GLSS$_a$, and **f** the average time for running each subsampling method once in Sect. 4.1

problem. We set the number of dimensions $d = 3, 10, 17$ and fix $N_{tr} = 5e5$, $N_{te} = 5e4$, $n = 300$. URS, SPlit, Twinning, DDS, and GLSS$_h$ are considered for subsampling in these cases. In GLSS$_h$, we reduce $d = 3, 10, 17$ to $d' = 2$ by SVD (named 2-d GLSS$_h$ hereafter) with the contributions accounting for 66.68%, 90.00%, 94.12% of the total variations in the training sets, respectively. We set $M = 881, 2477, 2731$ for the cases $d = 3, 10, 17$ in 2-d GLSS$_h$, respectively.

Based on the subsamples from these methods, linear regression and GP models are considered. Only the second-order terms with respect to x_1 and x_2, i.e., x_1^2, x_2^2, x_1x_2, are incorporated to fit the linear model and the mean function of the GP model. We also calculate the log RMSPEs of the fitted models based on the testing set and record the time cost of each subsampling method, which are shown in Fig. 4. It can be seen from Fig. 4a–c that the effect of DDS is still the worst in most cases, which implies that PCA almost cannot alleviate the problem caused by the dependence among coordinates for DDS. Other techniques should be taken into account for improving DDS. Twinning is worse than SPlit in many cases and shows instability. Compared with SPlit, 2-d GLSS$_h$ gives comparable and satisfying prediction performance in terms of the resulting log RMSPEs when d is small and it can be better than SPlit when d is large. As for the comparison of the computational complexity in Fig. 4d, SPlit still spends the most running time, whereas the time costs of

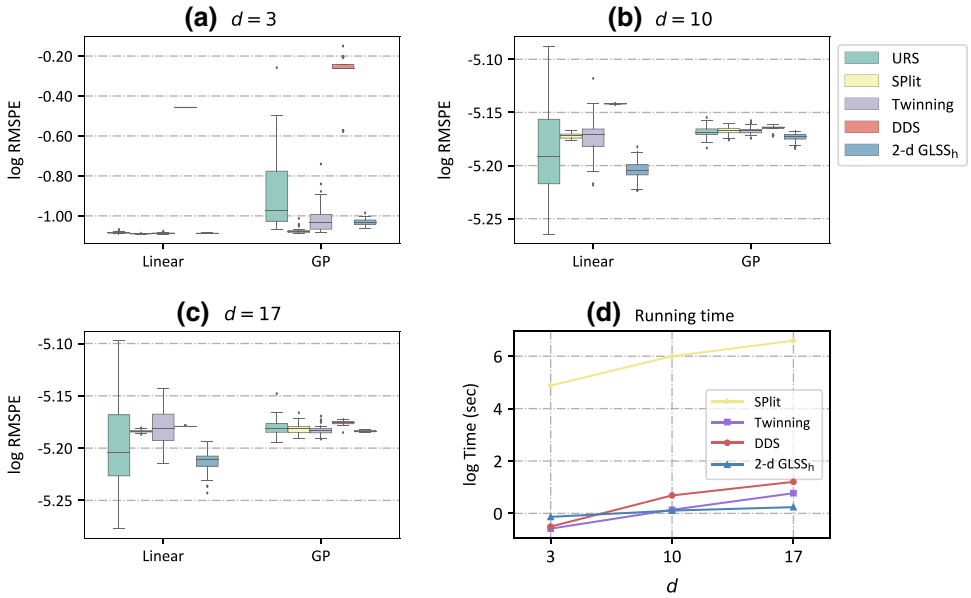

Fig. 4 The comparison of **a**–**c** the log RMSPEs for different numbers of dimension and models under URS, SPlit, Twinning, DDS, and 2-d GLSS_h and **d** the average time for running each subsampling method once in Sect. 4.2

Twinning, DDS, and 2-d GLSS_h are much lower than SPlit. When d goes larger, the cost of DDS mainly lies in finding the nearest neighbor in the d-dimensional data, while 2-d GLSS_h still conducts it in the 2-dimensional space. Thus, the complexity of 2-d GLSS_h becomes lower than DDS when d goes large. GLSS can also be faster than twinning with larger d. Hence, Fig. 4 demonstrates the effectiveness and speediness of 2-d GLSS_h for data with different original numbers of dimensions.

4.3 Comparison between GLSS_h and IBOSS, MV, MVc

In this section, we compare GLSS_h with some model-based methods, i.e., IBOSS, MV, and MVc, which derive the subsampling rules to optimize the estimators of the regression coefficients in their respective considered models. When the parameter of interest is the coefficient estimation, those model-based methods may outperform GLSS. However, with the sampled data, if our interest is changed, e.g., the population mean, GLSS may outperform those optimal subsampling methods for coefficient estimation in a specific model. To illustrate this point, we study two cases to compare GLSS_h with IBOSS, MV, and MVc. In Case 1, the data are generated from a linear model $y(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x} + \epsilon$; in Case 2, similar to Yu et al. (2022), the data are generated from a Poisson regression model such that given the predictor $\boldsymbol{x}$, the response y follows a Poisson distribution with mean $\text{E}(y|\boldsymbol{x}) = \exp(\boldsymbol{\beta}^\top \boldsymbol{x})$. For both cases, the sizes of the full data are $N = 5\text{e}5$ and the predictors in $\boldsymbol{x} = (x_1, \ldots, x_7)$ are generated from the uniform distribution, i.e., $x_1, x_5 \sim U(0, 1)$, $x_6, x_7 \sim U(-1, 1)$, $x_i = x_{i-1} + \epsilon_0$ for $i = 2, 3, 4$, and $\epsilon, \epsilon_0 \sim U(0, 0.1)$. The true value of $\boldsymbol{\beta}$ is set as a 7×1 vector of 0.5 and let $n = 500, 700, 900$.

In Case 1, we subsample from the full data by 2-d GLSS_h and IBOSS. In 2-d GLSS_h, the contribution rate with $d' = 2$ is 74.56% and we set $M = 907, 1202, 1277$ for $n = 500, 700, 900$, respectively. In Case 2, we subsample from the full data by 2-d GLSS_h, MV, and MVc. In 2-d GLSS_h, the contribution rate with $d' = 2$ is 71.45% and we set $M = 907, 1171, 1361$ for $n = 500, 700, 900$, respectively. We choose the subdata size as 200 to gain the pilot estimates in MV and MVc. For both cases, based on the n-size subsamples from the considered methods, we do regressions (linear regression for Case 1; Poisson regression for Case 2) and compute the average log MSE for the regression coefficient and the population mean. Figure 5 depicts the results for the two cases. In Fig. 5a and c, IBOSS, MV, and MVc show their superiority in coefficient estimation. It can also be seen that when $n = 500$, GLSS_h gives a relatively good performance, i.e., it has a similar log MSE value to IBOSS, and smaller log MSE than MVc. From Fig. 5b and d, compared with IBOSS, MV, and MVc, GLSS_h always has smaller log MSEs of the population mean whatever n is. Moreover, in the considered cases, MV and MVc always have larger log MSEs of the population mean than URS. The reason may be that the subsampling probabilities in MV and MVc are proportional to the values $|y_i - \exp(\hat{\boldsymbol{\beta}}_{\text{full}}^\top \boldsymbol{x}_i)|$ for $i = 1, \ldots, N$, where $\hat{\boldsymbol{\beta}}_{\text{full}}$ is the QLE based on the full data. Hence, it implies that the optimal subsampling methods with similar subsampling probabilities, e.g., Wang et al. (2018), can result in similar phenomena. In conclusion, when the evaluation criterion is inconsistent with the criterion that derives the subsampling rule, GLSS_h may be a better choice.

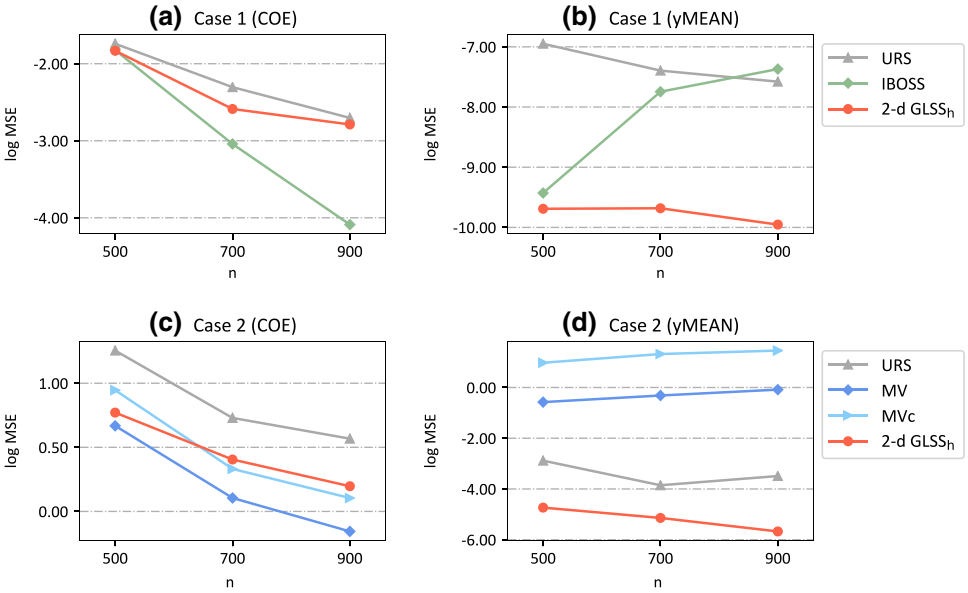

Fig. 5 The comparison of IBOSS, MV, MVc, and 2-d GLSS_h in terms of log MSE of the regression coefficient (COE) and the population mean (yMEAN) in Sect. 4.3

4.4 The effect of contribution rate in GLSS_h

In this subsection, we discuss the impact of the contribution rate (CR) based on SVD in GLSS_h according to a linear data-generating model as

$$y(\boldsymbol{x}) = x_1 + \cdots + x_{d-1} + \epsilon, \quad x_1, \ldots, x_{d-1} \overset{i.i.d}{\sim} \mathcal{N}(0,1), \epsilon \sim \mathcal{N}(0,1). \tag{4}$$

The number of dimensions d is set as 3, 4, 5, 8, $N_{\text{tr}} = 5\text{e}5$, $N_{\text{te}} = 5\text{e}4$ and $n = 300$. Both the training and testing sets are generated by the model (4). The numbers of dimension $d = 3, 4, 5, 8$ are reduced to $d' = 2$ by SVD with the contributions accounting for 93.87, 71.64, 57.90, 36.77% of the total variation in the training data. Contrary to Sect. 4.2, as the increase of d, the contribution rate becomes lower. We apply the 2-d GLSS_h to the training set, in which M is set as 881, 877, 883, 953 for $d = 3, 4, 5, 8$, respectively. Based on the subsamples, we also fit the linear and GP models incorporating the terms $x_1, \ldots, x_{d-1}$ and compute the log RMSPEs using the testing data based on the fitted models. From Fig. 6, it can be seen that with the decrease of the contribution rate, the variance of the resulting log RMSPEs based on 2-d GLSS_h would relatively increase compared with those based on URS. However, the prediction result based on 2-d GLSS_h is always better than URS with smaller log RMSPEs, even when the contribution rate is lower than 40%. It illustrates that with a high contribution rate, GLSS_h can attain good performance, and the corresponding performance would not be much worse when the rate is low.

4.5 The effective number of dimensions in GLSS_h

In the previous discussions, we mainly focus on the 2-d GLSS_h, i.e., we reduce the original number of dimensions d to $d' = 2$. In this subsection, we consider the impact of different d's with $d = 10$ by the data-generating model

$$y(\boldsymbol{x}) = x_1 + \cdots + x_5 + (x_6 + \cdots + x_9)^2 + \epsilon, \quad (x_1, \ldots, x_9)^\top \sim \mathcal{N}_9(\mathbf{0}, \boldsymbol{\Sigma}), \epsilon \sim \mathcal{N}(0,1), \tag{5}$$

where $\mathbf{0}$ is a 9×1 vector with zeros and the (i, j)th entry of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma}_{i,j} = 0.8^{|i-j|}$ for $i = 1, \ldots, 9$, $j = 1, \ldots, 9$. Here we also fix $N_{\text{tr}} = 5\text{e}5$, $N_{\text{te}} = 5\text{e}4$, $n = 300$ and set $d' = 2, 3, 4, 5$, whose contributions by SVD account for 70.97%, 80.22%, 87.72%, 91.73% of the total variation in the 10-dimensional training data, respectively. Then, the 2-d, 3-d, 4-d and 5-d GLSS_hs are performed with M set as 809, 4099, 9497, 34019, respectively. The linear, GP, and MARS models are considered based on the subsamples from the training set. For fitting the linear model and the mean function of the GP model, only the linear terms $x_1, \ldots, x_9$ are incorporated. Based on the fitted models and the testing set, we compute the log RMSPEs and compare them with the corresponding result under URS, which is shown in Fig. 7.

It can be seen that all the considered cases for GLSS_h are better than URS with smaller log RMSPEs. Especially, under the GP model that gives the best prediction result among the three models, 2-d GLSS_h even performs better than others. It implies that under the same original data, when the contribution rate is not low, e.g., higher than 70%, 2-d GLSS_h not only possesses the lowest computational complexity compared with other cases with larger d', but also retains a good performance, even better. For the cases $d' > 2$, better performance may exist compared with the current result in Fig. 7

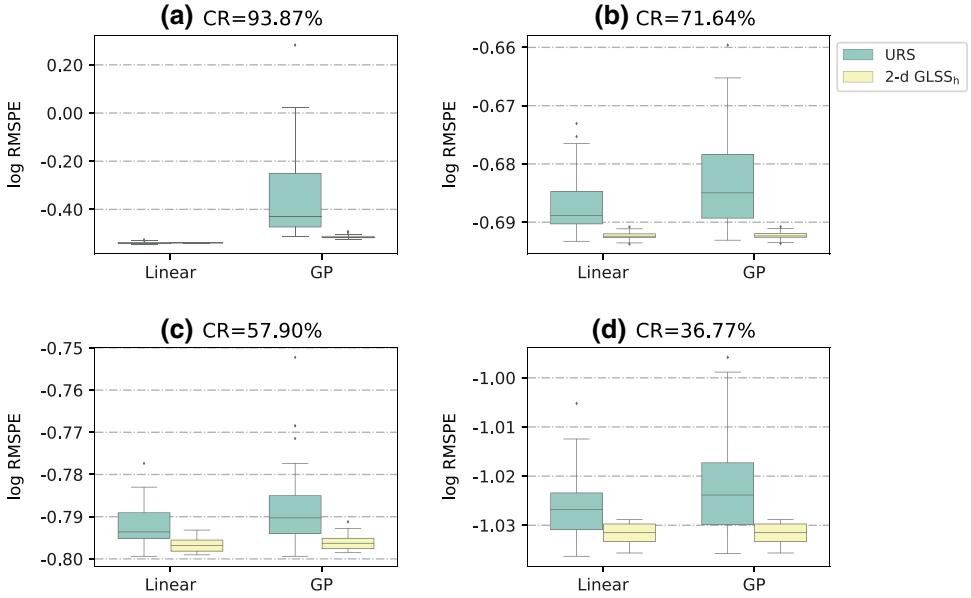

Fig. 6 The comparison of log RMSPEs under URS and 2-d $GLSS_h$ with different contribution rates (CRs) in Sect. 4.4

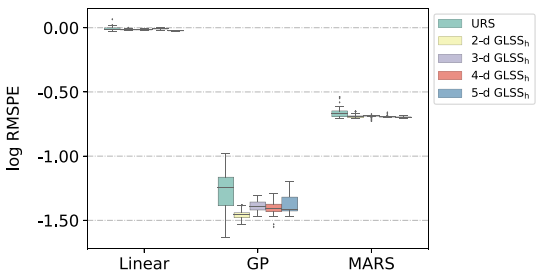

Fig. 7 The comparison of log RMSPEs under URS and $GLSS_h$ with $d' = 2, 3, 4, 5$ in Sect. 4.5

because of the higher contribution rate, but it may also need a larger M that further increases the complexity.

5 Real data studies

In this section, to further demonstrate the validity of GLSS, we apply it to six real datasets for data compression, where the datasets can be obtained from the UCI Machine Learning Repository (Dua and Graff 2019). Four of the datasets are used for regression and the other two are used for classification. In each dataset, we randomly partition the full data into a training set and a testing set. For comparison, URS, Twinning, and $GLSS_h$ are considered on the training set for subsampling. Similar to Sect. 4, we consider the impact of the size of the final subsample n on the subsampling performance under the first dataset. For other datasets, we fix n and mainly focus on the comparison of different subsampling methods. We use the default setting in the *twinning* R package to implement Twinning. For the datasets used for classification, there exist some categorical variables in the predictors or response. We transform these categorical variables into dummy variables, i.e., for a s-level categorical variable, we encode it using $s-1$ dummy variables. For example, if a categorical variable x has three levels, we need two dummy variables z_1, z_2, in which $x = 1$ corresponds to $z_1 = 1, z_2 = 0$; $x = 2$ corresponds to $z_1 = 0, z_2 = 1$; $x = 3$ corresponds to $z_1 = 0, z_2 = 0$. The dummy variables can be regarded as the numerical variables to perform GLSS and Twinning. All the numerical variables in these datasets are normalized to mean 0 and standard deviation 1 before subsampling. We repeat URS, Twinning, and $GLSS_h$ 50 times and in each repetition of $GLSS_h$, we slightly shift the GLP.

For regression, based on each subsample, we also perform the linear, GP, and MARS regressions. When the number of the predictors is larger than 5, we use the LASSO regression to replace the ordinary linear regression or select significant predictors by the Pearson correlation coefficients between the predictors and response before subsampling. We still adopt log RMSPE to measure the prediction performance on the testing set. For classification, we conduct the random forest (Breiman 2001) and the logistic regression on the obtained subsamples. For multi-class classification in logistic regression, we convert it into several binary classifications. Each classifier can predict the probability that a test sample belongs to the positive class in the prediction stage. Then we select the classifier with the highest probability and take its positive class as the prediction result. For both classification methods, the prediction performance is measured by the prediction accuracy, i.e., the number of correct predictions divided by the total number of predictions, in the testing set to compare the subsampling methods. Similar to Sect. 4, for the same training set, the parameter settings for each model under different subsamples are the same for a fair comparison.

5.1 Regression

5.1.1 Combined cycle power plant dataset

The combined cycle power plant (CCPP) dataset contains 9568 samples with 4 continuous predictors and the response is the net hourly electrical energy output of the plant. We partition the full data into a training set with 9000 samples and a testing set with 568 samples. For GLSS_h, we reduce the number of dimensions $d = 5$ to $d' = 2$ by SVD, which explains 84.65% of the total variation in the training data. We set $n = 300, 600, 900, 1200$ and perform URS, Twinning, 2-d GLSS_h with $M = 863, 1051, 1171, 1277$ for each n in turn. Figure 8 shows the result for the log RMSPEs based on the fitted models using different n-size subsamples. Clearly, in most cases of each subfigure in Fig. 8, the result under 2-d GLSS_h is more inclined to smaller log RMSPEs than those under URS and Twinning in the three regression methods. It implies that GLSS_h performs better than URS and Twinning in many cases. Moreover, it again illustrates that the subsample size n has little impact on the performance comparison.

5.1.2 Condition-based maintenance of naval propulsion plants dataset

The condition-based maintenance of naval propulsion plants (CBM) dataset contains 11,934 samples with 16 continuous predictors and 2 responses, that is, the gas turbine compressor and turbine decay state coefficients. We focus on the first response. The full data is partitioned into a training set with 10,000 samples and a testing set with 1934 samples. We reduce the dimension of the training data to 2 with the contribution accounting for 97.74% of the total variation. We fix $n = 300$ and perform URS, Twinning, and 2-d GLSS_h with $M = 929$. For modeling, we perform the LASSO regression, where the penalty coefficient is determined by 10-fold cross-validation, and model GP and MARS using the significant predictors by LASSO based on the subsamples. The result is shown in Fig. 9a. It can be seen that except for MARS, 2-d GLSS_h shows better prediction quality compared with URS and twinning. Under MARS, 2-d GLSS_h is still better than URS and slightly worse than twinning. However, 2-d GLSS_h has better worst-case performance than Twinning.

5.1.3 MiniBooNE particle identification dataset

The MiniBooNE particle identification (MBP) dataset has 130,065 samples with 50 particle ID real variables. We treat the last variable as the response and the other 49 variables as the predictors. The full data is partitioned into a training set with 120,000 samples and a testing set with 10,065 samples. Since the number of the dimensions is large, we first perform variable selection based on the normalized training data by calculating the Pearson correlation coefficients between the predictors and response.

It is found that the correlation is high and 8 significant predictors are selected. We further reduce the dimension from 9 (i.e., 8 significant predictors and 1 response) to $d' = 2$ with the contribution accounting for almost 100% of the total variation in the 9-dimensional data. We fix $n = 300$ and perform URS, Twinning, and 2-d GLSS_h with $M = 919$ on the reduced 9-dimensional data for data compression. Based on the obtained subsamples, we fit the response only using the significant variables by the considered modeling methods. The result is shown in Fig. 9b. Under MARS, 2-d GLSS_h is better than URS and Twinning. As for the results under LASSO and GP, all the log RMSPEs based on Twinning and 2-d GLSS_h are relatively small. 2-d GLSS_h performs better than URS, whereas it performs slightly worse than Twinning under LASSO and GP models. However, 2-d GLSS_h also has better worst-case performance under the GP model.

5.1.4 Wave energy converters dataset

The wave energy converters (WEC) dataset consists of 288,000 samples with 48 continuous predictors and 1 response, i.e., the total power output of the farm. The full data is divided into a training set with 250,000 samples and a testing set with 38,000 samples. Similar to the MBP dataset, we first perform variable selection based on the training data. 16 significant predictors are selected and then by SVD, we reduce the dimension from 17 (i.e., 16 significant predictors and 1 response) to 2 with the contribution accounting for 95.11% of the total variation. We fix $n = 300$ and URS, twinning, 2-d GLSS_h with $M = 911$ are performed on the reduced 17-dimensional data. For modeling, the number of significant predictors is still large and it would spend a large amount of time if all the significant predictors are incorporated into the model, especially for GP and MARS. Thus, we try to fit the models using partial significant predictors, which are selected according to the absolute values of the correlation coefficients. It is found that using the 7 most significant predictors can obtain relatively good prediction performance and the time cost for modeling is also reduced.

Figure 9c shows the corresponding prediction result. Under all the models, 2-d GLSS_h performs better than URS and Twinning with smaller log RMSPEs. Hence, based on the variable selection and the dimension reduction techniques, the effectiveness of 2-d GLSS_h is further illustrated for high-dimensional data.

5.2 Classification

5.2.1 High time resolution universe survey dataset

The high time resolution universe survey dataset (HTRU) contains 17,898 samples with 8 continuous predictors and

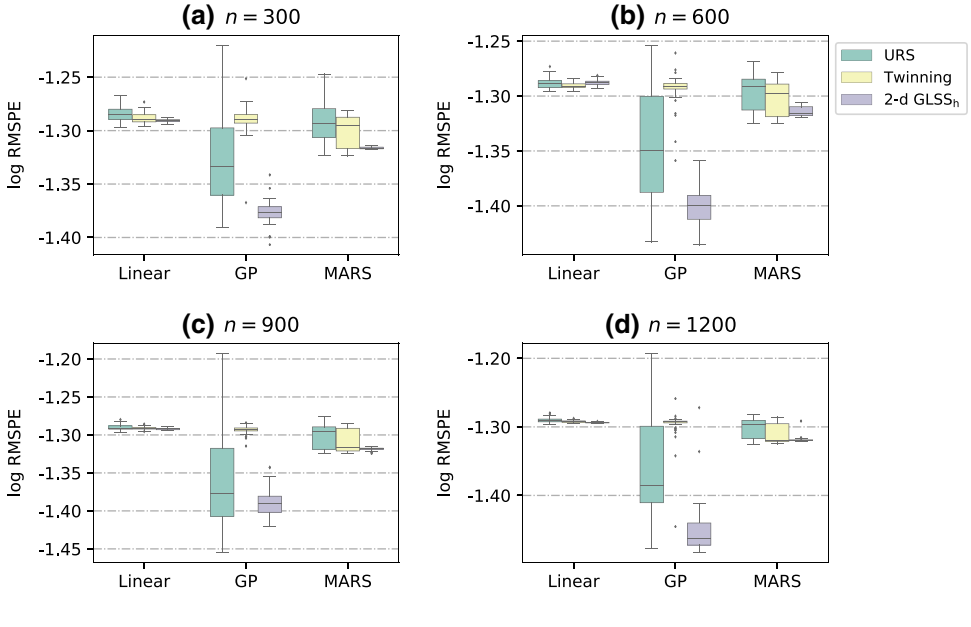

Fig. 8 The comparison of log RMSPEs under URS, Twinning, and 2-d $GLSS_h$ with $n = 300, 600, 900, 1200$ for the CCPP dataset in Sect. 5.1

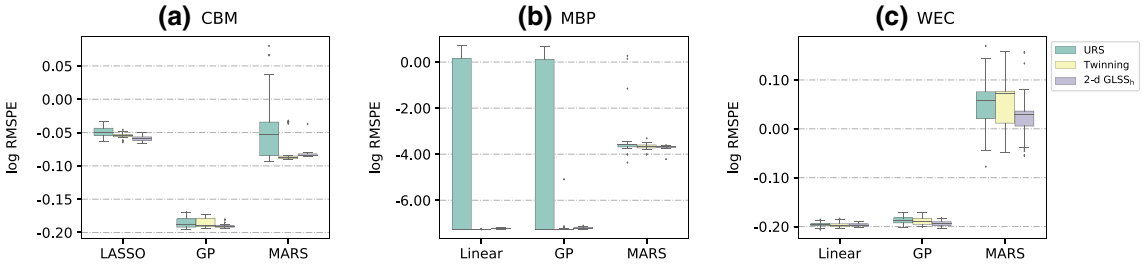

Fig. 9 The comparison of log RMSPEs under URS, Twinning, and 2-d $GLSS_h$ for the CBM, MBP, and WEC datasets in Sect. 5.1

a two-level categorical response. The categorical response is encoded with a dummy variable. Then, the full data is partitioned into a training set with 15,000 samples and a testing set with 2898 samples. We reduce the dimension of the training data from 9 to 2 with the contribution accounting for 77.69% of the total variation. We fix $n = 300$ and URS, Twinning, 2-d $GLSS_h$ with $M = 877$ are performed on the 9-dimensional training set for compression. Figure 10a shows the prediction accuracies on the testing set under different classification methods fitted by the obtained subsamples. It can be seen that Twinning is better than URS and the average prediction performance under 2-d $GLSS_h$ is better than those under both URS and Twinning in the two classification methods. Also, 2-d $GLSS_h$ has better worst-case performance than the other two subsampling methods.

5.2.2 Major atmospheric gamma imaging Cherenkov telescope dataset

The major atmospheric gamma imaging Cherenkov telescope dataset (MAGIC) consists of 19,020 samples with 10 continuous predictors and 1 categorical response. The response characterizes whether an event is a background or a signal and we encode the categorical response by a dummy variable. We partition the full data into a training set with 15,000 samples and a testing set with 4020 samples. Similar to the MBP and WEC datasets, we first conduct the variable selection based on the training set, in which 6 important predictors are selected. Then, we reduce the dimension from 7 (i.e., 6 important predictors and 1 dummy variable) to 2 with the contribution accounting for 71.67% of the total variation in the 7-dimensional data. We fix $n = 300$ and perform URS, Twinning, 2-d $GLSS_h$ with $M = 863$ on the reduced training data. Figure 10b shows the corresponding prediction result. Similarly, the performance of 2-d $GLSS_h$ is better than URS and Twinning with higher accuracy for the MAGIC dataset. It illustrates the effectiveness of $GLSS_h$ for classification when the categorical variables are encoded by dummy variables.

6 Conclusion and discussion

In this paper, we develop a model-free subsampling method called GLSS, which leverages the idea of the GLS sampling method. The convergence result of GLSS is derived based on the KDE technique and the GLS algorithm. A good setting

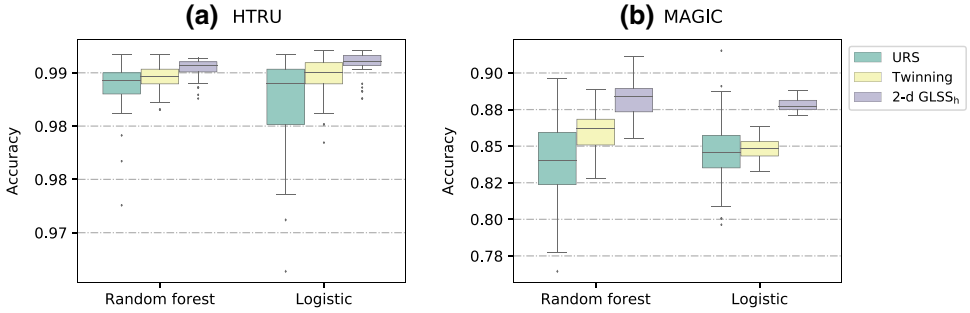

Fig. 10 The comparison of the prediction accuracy under URS and 2-d $GLSS_h$ for the HTRU and MAGIC datasets in Sect. 5.2

for the run size of the used space-filling design M is essential, which may be associated with the number of dimensions d, the size of the full data N, and the subsample size n. We recommend choosing $M = \exp(2(d + \nu))$, where $-d < \nu \leq 2$. The ν is larger when N or n is larger to capture the data structure better, while ν is smaller when d is larger for controlling the computational complexity. $GLSS_a$ is discussed to accelerate the algorithm by performing URS from the original large-scale data before running GLSS. We also propose $GLSS_h$ by utilizing SVD to extend GLSS to handle high-dimensional data.

In the simulation studies, under different data-generating models and modeling methods, we compare URS, SPlit, Twinning, DDS, GLSS, $GLSS_a$, and $GLSS_h$ with different subsample sizes, training data sizes, and different numbers of dimensions. The prediction result on the testing data and the running time show the superiority of GLSS, $GLSS_a$ and $GLSS_h$ on the model-free property. In most cases, compared with URS, Twinning, and DDS, GLSS or $GLSS_h$ gives better prediction performance with smaller log RMSPEs. Compared with SPlit, GLSS is comparable in many low-dimensional cases and $GLSS_h$ can be better in high-dimensional cases. It illustrates that when the contribution rate by SVD with d reduced to d' is not low, e.g., larger than 70%, d'-d $GLSS_h$ usually can produce good performance with relatively low computational complexity based on the high-dimensional data. $GLSS_a$ is effective when the data size is large, e.g., $N > 5e5$. The complexity of GLSS, $GLSS_a$, or $GLSS_h$ is also much lower than SPlit. Moreover, we also compare $GLSS_h$ with IBOSS, MV, and MVc. The simulation results show that $GLSS_h$ has a good performance and robustness when the evaluation interest is the mean population, which is different from the criterion that derives the subsampling rule for the model-based optimal subsampling methods.

In real data studies for data compression, both regression and classification problems are taken into consideration, where the categorical variables are encoded by dummy variables. In practice, there are usually not many significant variables and for ultrahigh-dimensional data, both the variable selection and dimension reduction techniques can be taken into account.

We reduce the number of all the variables or the significant variables by SVD with a relatively high contribution rate, i.e., all of them are larger than 70%. Then, we fit different models using the subsamples obtained by URS, Twinning, and $GLSS_h$ from the training data and test the prediction performance on the testing data. In most cases, $GLSS_h$ brings better prediction performance than URS and Twinning and significantly improves the worst-case performance. It further demonstrates that $GLSS_h$ is a model-robust and effective subsampling method. Hence, both the simulations and real data studies verify the validity of the proposed GLSS algorithm and its variants for data compression.

In addition, GLSS can also be extended to generate multiple batches of subsamples by performing random shifts on the space-filling design such that each batch of the subsample is a statistically good representation of the full data. A possible application is to replace the uniform random sample in each iteration of the stochastic gradient descent algorithm (Bottou 2012) with the subsample from each batch of GLSS, which may bring a higher convergence rate and more stable performance. The deterministic shifts for the space-filling design can also be studied for a better space-filling property. Another potential application is in the scalable Markov chain Monte Carlo for more representative subsampling in some sense. It is beyond the scope of the paper but important and worthy of further investigation.

Acknowledgements The authors would like to thank the two referees for their valuable comments that lead to a significant improvement of the presentation. This work was partial supported by the National Natural Science Foundation of China (11871288 and 12131001)), the Fundamental Research Funds for the Central Universities, LPMC, and KLMDASR.

Author Contributions Yi, S. wrote the main manuscript text and Zhou, Y. provided the research idea and modified the whole paper. All authors reviewed the manuscript.

Declarations

Conflict of interest The authors declare no competing interests.

Appendix

Proof of Lemma 1 By Theorem 2 in Jiang (2017), we directly have

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |\hat{f}_{\mathrm{K}}(\boldsymbol{\theta}) - f(\boldsymbol{\theta})| = O_{\mathrm{P}}(\Delta).$$

Let the support of f be $\mathcal{T}$ and the volume of $\mathcal{T}$ is $v(\mathcal{T}) = V_T < \infty$. Then,

$$\begin{aligned}\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |\hat{F}_{\mathrm{K}}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| &= \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \left| \int_{(-\infty,\boldsymbol{\theta}]\cap\mathcal{T}} (\hat{f}_{\mathrm{K}}(\boldsymbol{\vartheta}) - f(\boldsymbol{\vartheta}))\mathrm{d}\boldsymbol{\vartheta} \right| \\ &\le \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \int_{(-\infty,\boldsymbol{\theta}]\cap\mathcal{T}} |\hat{f}_{\mathrm{K}}(\boldsymbol{\vartheta}) - f(\boldsymbol{\vartheta})|\mathrm{d}\boldsymbol{\vartheta} \\ &\le V_T \cdot \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |\hat{f}_{\mathrm{K}}(\boldsymbol{\theta}) - f(\boldsymbol{\theta})| = O_{\mathrm{P}}(\Delta),\end{aligned}$$

which completes the proof. □

Proof of Lemma 2 Let $\tilde{F}(\boldsymbol{\theta})$ be the multinomial distribution placing mass $\hat{f}_{\mathrm{K}}(\boldsymbol{s}_k)/\sum_{i=1}^M \hat{f}_{\mathrm{K}}(\boldsymbol{s}_i)$ at $\boldsymbol{s}_k, k = 1,\ldots,M$. Similar to the proof of Lemma 1(1) in Yi et al. (2022), if $\mathcal{S}$ is the low-discrepancy point set and $\hat{f}_{\mathrm{K}}(\boldsymbol{x})I_{[-\infty,\boldsymbol{\theta}]}(\boldsymbol{x})$ is of uniformly bounded variation for any $\boldsymbol{\theta} \in \mathbb{R}^d$, then for any $\boldsymbol{\theta} \in \mathbb{R}^d$, we have

$$|\tilde{F}(\boldsymbol{\theta}) - \hat{F}_{\mathrm{K}}(\boldsymbol{\theta})| = O(\kappa(M)).$$

Let $D_{\mathrm{m}} = \{(2j-1)/2n : j = 1,\ldots,n\}$. Then, by the proof of Theorem 2 in Yi et al. (2022), we can obtain that

$$F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{u\in D_{\mathrm{m}}} \varrho_{\boldsymbol{\theta}}(u), \quad \tilde{F}(\boldsymbol{\theta}) = \int_{[0,1]} \varrho_{\boldsymbol{\theta}}(u)\mathrm{d}u.$$

Thus, $|F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta}) - \tilde{F}(\boldsymbol{\theta})|$ can be viewed as the quasi-Monte Carlo error based on D_{m}. The star discrepancy of D_{m} is always no more than $1/n$. Hence, by the Koksma-Hlawka inequality (Hlawka 1961), if $\varrho_{\boldsymbol{\theta}}(u)$ is of uniformly bounded variation for any $\boldsymbol{\theta} \in \mathbb{R}^d$ and run size M of the space-filling design $\mathcal{S}$, then for any $\boldsymbol{\theta} \in \mathbb{R}^d$, we have

$$|F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta}) - \tilde{F}(\boldsymbol{\theta})| = O\left(\frac{1}{n}\right).$$

Since the above formula is valid for any $\boldsymbol{\theta} \in \mathbb{R}^d$, it is also true for the supremum value. Then, by the triangle inequality,

$$\begin{aligned}&\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - \hat{F}_{\mathrm{K}}(\boldsymbol{\theta})| \\ &\le \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |\tilde{F}(\boldsymbol{\theta}) - \hat{F}_{\mathrm{K}}(\boldsymbol{\theta})| + \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta}) - \tilde{F}(\boldsymbol{\theta})| \\ &= O\left(\max\left\{\kappa(M), \frac{1}{n}\right\}\right),\end{aligned}$$

which completes the proof.

Proof of Theorem 3 Based on Lemmas 1 and 2, by the triangle inequality, we directly have

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F_{\tilde{\mathcal{Z}}_{\mathrm{s}}}(\boldsymbol{\theta}) - F(\boldsymbol{\theta})| = O_{\mathrm{P}}\left(\max\left\{\Delta, \kappa(M), \frac{1}{n}\right\}\right).$$

Let $\mathcal{D} = \otimes_{i=1}^d [a_i, b_i]$ and δ be the volume of the union of the balls $\mathcal{B}(\boldsymbol{s}_i, r), i = 1,\ldots,M$, divided by the volume of $\otimes_{i=1}^d [a_i - r, b_i + r]$. Since r is set as half of the separation distance of $\mathcal{S}$, we have $\delta\prod_{i=1}^d (2r + b_i - a_i) = Mv_d r^d$. Without loss of generality, let $a_i = 0$ and $b_i = 1$ for $i = 1,\ldots,d$, which leads to $r = 1/[(Mv_d\delta^{-1})^{1/d} - 2]$. Then, we have $r = O(M^{-1/d})$, which is also true when a_i and $b_i (a_i < b_i)$ take any other bounded values for $i = 1,\ldots,d$.

Based on the result, we have $\Delta = O(M^{-\alpha/d} + (M\log N/N)^{1/2})$. In addition, we can easily obtain that $M^{-\alpha/d} > 1/M$ if $d = 1$, and $M^{-\alpha/d} > (\log M)^d/M$ as $M \to \infty$ if $d > 1$. Thus, $O_{\mathrm{P}}(\max\{\Delta, \kappa(M)\}) = O_{\mathrm{P}}(M^{-\alpha/d} + (M\log N/N)^{1/2})$ as $M \to \infty$, which completes the proof.

Proof of Lemma 4 For any $z_{\mathrm{s}} \in \mathcal{Z}_{\mathrm{s}}$ and $\boldsymbol{\theta} \in \mathbb{R}^d$, we have

$$\begin{aligned}\mathrm{P}(z_{\mathrm{s}} \in (-\infty, \boldsymbol{\theta}]) &= \mathrm{E}(\mathrm{E}(I_{(-\infty,\boldsymbol{\theta}]}(z_{\mathrm{s}} \mid \mathcal{Z}))) = \mathrm{E}(F_{\mathcal{Z}}(\boldsymbol{\theta})) \\ &= F(\boldsymbol{\theta}),\end{aligned}$$

where $F_{\mathcal{Z}}$ is the ECDF of $\mathcal{Z}$. Hence, any element in $\mathcal{Z}_{\mathrm{s}}$ follows the CDF F. By the multivariate Dvoretzky-Kiefer-Wolfowitz inequality (Naaman 2021), we have

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} |F(\boldsymbol{\theta}) - F_{\mathcal{Z}_{\mathrm{s}}}(\boldsymbol{\theta})| = O_{\mathrm{P}}\left(\frac{1}{\sqrt{N_{\mathrm{s}}}}\right),$$

which completes the proof.

References

Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)

Bottou, L.: Stochastic Gradient Descent Tricks, Neural Networks: Tricks of the Trade, pp. 421–436. Springer, Berlin (2012)

Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

Davis, R.A., Lii, K.S., Politis, D.N.: Remarks on Some Nonparametric Estimates of a Density Function, Selected Works of Murray Rosenblatt, pp. 95–100. Springer, New York (2011)

Dua, D., Graff, C.: UCI machine learning repository. URL http://archive.ics.uci.edu/ml (2019)

Fan, J., Lv, J.: Sure independence screening for ultrahigh dimensional feature space. J. R. Stat. Soc. Ser. B **70**(5), 849–911 (2008)

Fang, K.T., Liu, M.Q., Qin, H., Zhou, Y.D.: Theory and Application of Uniform Experimental Designs. Springer, Singapore (2018)

Friedman, J.H.: Multivariate adaptive regression splines. Ann. Stat. **19**(1), 1–67 (1991)

Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109 (1970)
Hlawka, E.: Funktionen von beschränkter variatiou in der theorie der gleichverteilung. Annali di Matematica Pura ed Applicata **54**(1), 325–333 (1961)
Jiang, H.: Uniform convergence rates for kernel density estimation. In: International Conference on Machine Learning, pp. 1694–1703. PMLR (2017)
Joseph, V.R., Vakayil, A.: Split: an optimal method for data splitting. Technometrics **64**(2), 166–176 (2022)
Liang, F., Wong, W.H.: Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. J. Am. Stat. Assoc. **96**(454), 653–666 (2001)
Ma, P., Mahoney, M., Yu, B.: A statistical perspective on algorithmic leveraging. In: International Conference on Machine Learning, pp. 91–99. PMLR (2014)
Mahoney, M.W.: Randomized algorithms for matrices and data. Found. Trends Mach. Learn. **3**(2), 123–224 (2011)
Maire, F., Friel, N., Alquier, P.: Informed sub-sampling MCMC: approximate Bayesian inference for large datasets. Stat. Comput. **29**(3), 449–482 (2019)
Meng, C., Xie, R., Mandal, A., Zhang, X., Zhong, W., Ma, P.: Lowcon: a design-based subsampling approach in a misspecified linear model. J. Comput. Graph. Stat. **30**(3), 694–708 (2021)
Naaman, M.: On the tight constant in the multivariate Dvoretzky–Kiefer–Wolfowitz inequality. Stat. Probab. Lett. **173**, 109088 (2021)
Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Math. Comput. **75**(254), 903–920 (2006)
Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**(3), 1065–1076 (1962)
Rubin, D.B.: A noniterative sampling/importance resampling alternative to data augmentation for creating a few imputations when fractions of missing information are modest: the sir algorithm. J. Am. Stat. Assoc. **82**, 544–546 (1987)
Santner, T.J., Williams, B.J., Notz, W.I., Williams, B.J.: The Design and Analysis of Computer Experiments. Springer, New York (2003)
Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B **58**(1), 267–288 (1996)
Ting, D., Brochu, E.: Optimal subsampling with influence functions. In: Advances in Neural Information Processing Systems, pp. 3650–3659 (2018)
Vakayil, A., Joseph, V.R., Mak, S., Vakayil, M.A.: Package 'SPlit'. R package version 1.0 (2022)
Vakayil, A., Joseph, V.R., Vakayil, M.A.: Package 'twinning'. R package version 1.0 (2022)
Vakayil, A., Joseph, V.R.: Data twinning. Stat. Anal. Data Min. ASA Data Sci. J. **15**, 598–610 (2022)
Wang, D., Joseph, V.R.: Mined: minimum energy design. R Package Version 1.0 (2022)
Wang, Y.C., Ning, J.H., Zhou, Y.D., Fang, K.T.: A new sampler: randomized likelihood sampling. In: Souvenir Booklet of the 24th International Workshop on Matrices and Statistics, pp. 255–261 (2015)
Wang, H., Ma, Y.: Optimal subsampling for quantile regression in big data. Biometrika **108**(1), 99–112 (2021)
Wang, H., Zhu, R., Ma, P.: Optimal subsampling for large sample logistic regression. J. Am. Stat. Assoc. **113**(522), 829–844 (2018)
Wang, H., Yang, M., Stufken, J.: Information-based optimal subdata selection for big data linear regression. J. Am. Stat. Assoc. **114**(525), 393–405 (2019)
Wang, L., Elmstedt, J., Wong, W.K., Xu, H.: Orthogonal subsampling for big data linear regression. Ann. Appl. Stat. **15**(3), 1273–1290 (2021)
Yi, S.Y., Liu, Z., Liu, M.Q., Zhou, Y.D.: Global likelihood sampler for multimodal distributions. Submitted (2022)
Yu, J., Wang, H., Ai, M., Zhang, H.: Optimal distributed subsampling for maximum quasi-likelihood estimators with massive data. J. Am. Stat. Assoc. **117**(537), 265–276 (2022)
Zhang, M., Zhou, Y.D., Zhou, Z., Zhang, A.J.: Model-free subsampling method based on uniform designs. arXiv:2209.03617 (2022)