INFORMS Journal on Data Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Diversity Subsampling: Custom Subsamples from Large Data Sets

Boyang Shang, Daniel W. Apley, Sanjay Mehrotra

To cite this article:
Boyang Shang, Daniel W. Apley, Sanjay Mehrotra (2023) Diversity Subsampling: Custom Subsamples from Large Data Sets. INFORMS Journal on Data Science 2(2):161-182. https://doi.org/10.1287/ijds.2022.00017

https://pubsonline.informs.org/journal/ijds

INFORMS JOURNAL ON DATA SCIENCE
Vol. 2, No. 2, October–December 2023, pp. 161–182
ISSN 2694-4022 (print), ISSN 2694-4030 (online)

Diversity Subsampling: Custom Subsamples from Large Data Sets

Boyang Shang,[a] Daniel W. Apley,[a,*] Sanjay Mehrotra[a]

[a] Industrial Engineering and Management Science, Northwestern University, Evanston, Illinois 60208
*Corresponding author
Contact: boyangshang2015@u.northwestern.edu, https://orcid.org/0000-0001-5379-6880 (BS); apley@northwestern.edu, https://orcid.org/0000-0002-8545-4612 (DWA); mehrotra@northwestern.edu, https://orcid.org/0000-0003-1106-1901 (SM)

Received: June 13, 2022
Revised: June 28, 2023
Accepted: September 20, 2023
Published Online in Articles in Advance: November 22, 2023

https://doi.org/10.1287/ijds.2022.00017

Abstract. Subsampling from a large unlabeled (i.e., no response values are available yet) data set is useful in many supervised learning contexts to provide a global view of the data based on only a fraction of the observations. In this paper, we borrow concepts from the well-known sampling/importance resampling technique, which samples from a specified probability distribution, to develop a diversity subsampling approach that selects a subsample from the original data with no prior knowledge of its underlying probability distribution. The goal is to produce a subsample that is independently and uniformly distributed over the support of distribution from which the data are drawn, to the maximum extent possible. We give an asymptotic performance guarantee of the proposed method and provide experimental results to show that the proposed method performs well for typical finite-size data. We also compare the proposed method with competing diversity subsampling algorithms and demonstrate numerically that subsamples selected by the proposed method are closer to a uniform sample than subsamples selected by other methods. The proposed diversity subsampling (DS) algorithm is more efficient than known methods. It takes only a few minutes to select tens of thousands of subsample points from a data set of size one million. Our DS algorithm easily generalizes to select subsamples following distributions other than uniform. We provide a Python package (FADS) that implements the proposed method.

History: Kwok-Leung Tsui served as the senior editor for this article.
Funding: This work was supported by the National Science Foundation [Grant CMMI-1436574], Northwestern University, the Advanced Research Projects Agency-Energy, and the U.S. Department of Energy [Award DE-AR0001209].
Data Ethics & Reproducibility Note: No data ethics considerations are foreseen related to this article. The code capsule is available on Code Ocean at https://doi.org/10.24433/CO.8309237.v3 and in the e-Companion to this article (available at https://doi.org/10.1287/ijds.2022.00017).

Keywords: diversity subsampling • custom subsampling • representative • space-filling • fully sequential

1. Introduction

Diversity subsampling selects a subset of points from a data set with the goal of having the selected points spread out evenly over the data space. The goal of diverse samples is ubiquitous in the field of experimental design and is a common design criterion. In the specific situation in which some supervised learning model will be fit to the selected subsample (after some follow-up experiment in which response labels are observed for each point in the sample), a uniform or space-filling subsample will typically be beneficial. For instance, training the model with a space-filling subsample, as opposed to a random subsample that follows the same distribution as the training data, provides some robustness to the covariate drift issue (i.e., the distribution of the future test data to which the supervised learning model will be applied is different from the training data distribution). As another example, sometimes the goal may be to use the fitted supervised learning algorithm to optimize the response (regarded as a function of the covariates), a more space-filling subsample seems advantageous because it reveals the behavior of the response in the entire covariate space. In the literature, diversity subsampling has been found useful in a variety of situations, for example, Yu and Kim (2010), Puzyn et al. (2011), Haussmann et al. (2020), Song et al. (2022b), and Silveira and Barbeira (2022), to name a few.

Suppose we have data set D consisting of an identically and independently distributed (i.i.d.) sample of size N of some random vector $\boldsymbol{X} \in \mathbb{R}^q$; that is, $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in \mathbb{R}^q$. Let $\mathcal{S}_n = \{\boldsymbol{x}_{j_1}, \ldots, \boldsymbol{x}_{j_n}\} \subset D$, for $\{j_1, \ldots, j_n\} \subset \{1, 2, \ldots, N\}$ denote a subsample of size n selected from D. In the literature, loosely speaking, $\mathcal{S}_n$ is called a diverse subsample from D if the points it constitutes are as different from each other as possible (sometimes called the repulsiveness property; Wang et al.

2018, Biyik et al. 2019) and cover as much of the effective support of the data as possible. In some machine learning applications, each observation $\boldsymbol{x}_i$ is already labeled with a response $\boldsymbol{y}_i$; our paper focuses on the setting where the labels $\{\boldsymbol{y}_i\}_{i=1}^N$ are not available, and one will observe the responses only for the n cases in $\mathcal{S}_n$ (e.g., by conducting some follow-up experimental or observational study, surveying the cases, etc.) which will then serve as the training data for fitting some supervised learning model. For situations in which all N response labels are available and the goal is supervised compression of the data (Joseph and Mak 2021, Chen and Zhang 2022).

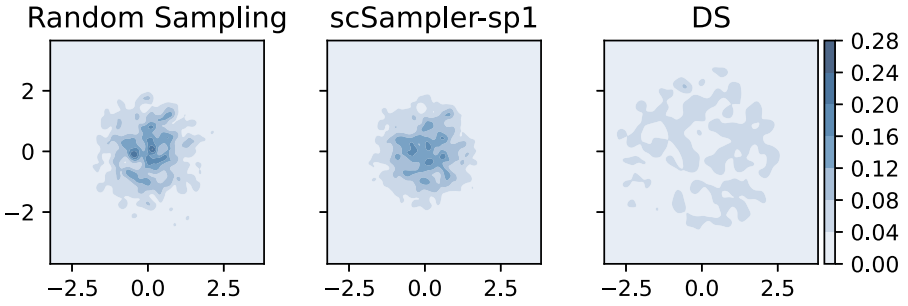
Existing popular methods for diversity subsampling from given data include determinantal point processes (DPP; Biyik et al. 2019), computer-aided design of experiments (CADEX; Kennard and Stone 1969), Poisson disk sampling (PDS; Cook 1986, Yuksel 2015), and k-means clustering (MacQueen et al. 1967, Wu 2018). DPP can be used for diversity subsampling by first specifying an appropriately constructed kernel matrix so that the most diverse set of points will have the largest probability of being selected under this DPP; this set is thus the mode of the DPP. It has been shown that finding this mode is an NP-hard problem (Ko et al. 1995). Various algorithms have been proposed to approximate the mode of a DPP (Biyik et al. 2019, Han and Gillenwater 2020). CADEX selects points sequentially from data such that the distance between each newly selected point and its closest point in the previously selected point set is as large as possible. Song et al. (2022b) recently proposed an efficient heuristic for the CADEX algorithm, which is called the scSampler algorithm. PDS is a conceptually simple algorithm that selects subsequent points outside of the union of hyper-spherical neighborhoods of all selected points. The radius of these hyper-spherical neighborhoods is usually user specified and taken as an input of the algorithm, which ensures that the pairwise distances between all selected points are no smaller than twice the specified radius (Cook 1986). Existing PDS algorithms that do not require the users to prespecify a fixed radius include McCool and Fiume (1992) and Yuksel (2015). Wu (2018) proposed the RD ALR method (representativeness and diversity, active learning for regression) that uses k-means clustering (MacQueen et al. 1967) to select a diverse subsample from D; it selects each point in $\mathcal{S}_n$ sequentially. At iteration j, RD ALR clusters the entire data into $j+1$ clusters using k-means, and the newly selected point are chosen from clusters that do not contain any previously selected points.

One common property of the previously mentioned methods is that the subsamples they select are all repulsive, in the sense that they attempt to ensure that the selected points are as far away from each other as possible. Although repulsiveness is an appealing property of a uniform or space-filling subsample, it might not be achievable when there are many replicated observations in D. In contrast to the existing subsampling algorithms, we define our diversity subsampling goal as to select a subsample that follows, as closely as possible, a mutually independent uniform distribution over the effective support of D, and we refer to our proposed algorithm as the diversity subsampling (DS) algorithm. Our DS algorithm has substantial benefits compared with the existing ones. First, aiming to select an i.i.d uniform subsample over the effective support of D allows replications to occur in the subsample. This has advantages in applications in which the response variable Y is observed with error and/or when there are additional latent variables because in these situations, it may be desirable to observe Y at multiple very similar X values. Second, our DS algorithm handles data with large numbers of replicated values along all or certain dimensions much better than existing algorithms: In this challenging situation, the subsample selected by the DS algorithm is still as evenly spread out over the data space as possible, but the subsamples selected by existing algorithms degrade to random sampling (we provide an example to illustrate this shortly). Third, our DS algorithm turns out to be much more computationally efficient, especially with larger N and n (which is increasingly common in the big data era) than existing methods (see Section 5.4 for details). Last, it is easy to generalize our DS algorithm to select subsamples following any desired distribution (not just uniform).

To illustrate the better uniformity properties of the DS subsample than existing methods when there are many replicated observations in D, consider the following simple example in which $\boldsymbol{X}$ follows a bivariate standard normal distribution with independent components (see Section 5.3 for experimental results using quantitative measures with additional examples). We set $q=2$, $N=$ 10,000, and $n=2{,}000$ in this example. The 10,000 data points were generated using 2,000 independently drawn points from the bivariate standard normal distribution, each of which was replicated five times. Figure 1 compares heat maps of the estimated density of typical subsamples selected by three different methods: random sampling from a data set, scSampler-sp1 (Song et al. 2022b), and our DS algorithm (see Section 2 for details of the DS algorithm). The densities were estimated using a Gaussian kernel with a bandwidth of 0.1. As Figure 1 shows, the points selected by random sampling (the left plot of Figure 1) follow the original bivariate standard normal distribution; that is, they concentrate in the middle region of the plotted area, where the data points (not shown in Figure 1) are denser. The subsample selected by the scSampler-sp1 algorithm (the middle plot of Figure 1) also exhibits the same phenomenon. In contrast, the subsample selected by DS (the right plot of Figure 1) is much closer to being uniformly distributed over the effective support of the data.

The structure of this paper is as follows. Section 2 presents our DS algorithm. Section 3 discusses generalizations of the proposed DS algorithm to select subsamples

Figure 1. (Color online) Heat Maps Comparing the Estimated Densities of the Selected Subsamples Using Three Different Methods for $q = 2$



Notes. The left, middle, and right plots are for the subsamples selected by random sampling, scSampler-sp1, and DS, respectively. The DS subsample is far more uniformly distributed than the other two for this data set consisting of many replicated observations.

following any desired distribution. Section 4 discusses theoretical convergence of the sample produced by the DS algorithm to the desired target distribution. Section 5 numerically demonstrates the effectiveness of the DS algorithm and its advantages compared with known methods using data exhibiting a variety of distributions. Section 6 concludes the paper.

2. DS Algorithm for Diversity Subsampling

The goal of our DS algorithm is to select a subsample $\mathcal{S}_n$ of size n (typically, n is much less than the data size N) points in $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ that is distributed as closely as possible to an i.i.d. sample from the uniform distribution over S, the effective support of the data distribution. First, we discuss the case when the distribution of $\boldsymbol{X}$ is continuous, and then we extend our DS method for general data sets with discrete or mixed continuous and discrete data distributions. Let $f(\boldsymbol{x})$ and S denote the probability density function (p.d.f) of $\boldsymbol{X}$ and its support, both of which are assumed unknown. We develop two versions of the DS algorithm, for sampling from D with or without replacement. The former is relevant in a smaller class of applications in which it is desirable to observe the response Y multiple times for the same observation $\boldsymbol{X}$. For brevity, we focus on the sampling-without-replacement version of the DS algorithm (denoted as the DS algorithm) in this section and discuss the sampling-with-replacement version (denoted as the DS-WR algorithm) briefly. We present certain asymptotic performance results for the DS algorithm in a more general way in Section 4, where we demonstrate that under certain assumptions the uniformity and independence of points in $\mathcal{S}_n$ can be guaranteed asymptotically.

Our approach adapts the well-known sampling/importance resampling (SIR) algorithm (Rubin 1987, 1988), which subsamples proportional to certain weights from simulated data generated from some proposal distribution (as opposed to subsampling from a given data set). The SIR algorithm works as follows. To generate i.i.d. samples from a target distribution with p.d.f. $g(\boldsymbol{x})$, SIR samples from some proposal p.d.f. $f_{\text{prop}}(\boldsymbol{x})$ that is ideally similar to g and convenient to sample from. SIR first generates a sample from f_{prop}, say $D_{\text{SIR}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$, and then randomly samples (with or without replacement) a smaller set $\mathcal{S}_{\text{SIR}}$ from D_{SIR} with the probability of each point being selected proportional to $g(\boldsymbol{x}_i)/f_{\text{prop}}(\boldsymbol{x}_i)$, $i = 1, \ldots, M$. The selected $\mathcal{S}_{\text{SIR}}$ is asymptotically an i.i.d. sample from $g(\boldsymbol{x})$ as M approaches infinity (Skare et al. 2003).

The SIR algorithm subsamples from a simulated data set generated from a known proposal distribution and is not directly applicable to subsample from a given data set with unknown distribution. We adapt it to the latter setting, as follows. If we view the (unknown) data distribution $f(\boldsymbol{x})$ as the proposal distribution and the uniform distribution over S as the target distribution, then we can obtain an approximately i.i.d. uniform distribution over S by subsampling from D with the probability of each point being selected proportional to $(1/|S|)/f(\boldsymbol{x}_i) \propto 1/f(\boldsymbol{x}_i) \approx 1/\hat{f}_N(\boldsymbol{x}_i)$, for $i = 1, \ldots, N$, for some suitable estimator $\hat{f}_N$ of the density f. Procedure 1 summarizes the DS algorithm for sampling without replacement. In Section 4, we present theoretical convergence results for the DS algorithm with known $f(\boldsymbol{x})$, and in Section 5, we demonstrate numerically that it produces samples that reasonably approximate the desired target distribution. In the DS algorithm, we first estimate $f(\boldsymbol{x})$ at each data point in D (see Appendix B for details). Denote the estimated density at $\boldsymbol{x}_i$ as $\hat{f}(\boldsymbol{x}_i)$, for $i = 1, \ldots, N$. Then the DS algorithm selects the n points in $\mathcal{S}_n$ from D sequentially at each iteration k as follows.

At iteration $k = 1$, the DS algorithm selects the index $j_1 \in \{1, \ldots, N\}$ of the first point in $\mathcal{S}_n$, denoted by $\boldsymbol{x}_{j_1}$, with probability

$$P(j_1) = \frac{\frac{1}{\hat{f}(\boldsymbol{x}_{j_1})}}{\sum_{i=1}^{N} \frac{1}{\hat{f}(\boldsymbol{x}_i)}}. \tag{1}$$

At iteration $k \geq 2$, the DS algorithm selects the index $j_k \in \{1, \ldots, N\} \setminus \{j_1, \ldots, j_{k-1}\}$ of the next sampled point $\boldsymbol{x}_{j_k}$

with probability

$$P(j_k|j_1,\dots,j_{k-1}) = \frac{\frac{1}{\hat{f}(\boldsymbol{x}_{j_k})}}{\sum_{i=1}^{N}\frac{1}{\hat{f}(\boldsymbol{x}_i)} - \sum_{i=1}^{k-1}\frac{1}{\hat{f}(\boldsymbol{x}_{j_i})}}. \tag{2}$$

Three practical issues must be addressed when applying the previous idea to real data sets. One issue is that n might not be negligibly small compared with N. Because we are sampling without replacement, the size of the remaining data set $D \setminus \mathcal{S}_{k-1}$ at iteration k decreases with k; thus the distribution of the remaining data can change dramatically from the distribution of the original data D. We remedy this by updating the estimated density regularly so that it reflects the distribution of the remaining data and not the distribution of D. In particular, we update the density of remaining points in D after every $M = \max\{100, \lfloor n/\mathcal{U} \rfloor\}$ points have been selected, where $\mathcal{U}$ is a user-chosen integer (we use $\mathcal{U} = 10$ in all examples). Here the symbol $\lfloor n/\mathcal{U} \rfloor$ denotes the largest integer not larger than $n/\mathcal{U}$. We provide a method to update density efficiently in Appendix B.

The second issue is that the density estimation method proposed in Appendix B only applies when the data distribution is continuous. In the case that the data distribution is discrete or mixed continuous and discrete, we approximate the true distribution with a continuous one by adding a small Gaussian noise (lines 3–10 of Procedure 1). For simplicity and because data are often rounded, we apply this Gaussian perturbation for all data sets regardless of the true data distribution being discrete or not. To be specific, for the density estimation step, $\forall i \in \{1,\dots,N\}$, we replace $\boldsymbol{x}_i$ with $\boldsymbol{x}_i + \sigma_p \boldsymbol{Z}_i$, where $\{\boldsymbol{Z}_i\}_{i=1}^N$ follow i.i.d. multivariate normal distribution $\mathcal{N}(\boldsymbol{0}_q, \boldsymbol{1}_{q\times q})$, where $\boldsymbol{0}_q$ is the zero vector in $\mathcal{R}^q$ and $\boldsymbol{1}_{q\times q}$ is the identity matrix in $\mathcal{R}^{q\times q}$. We choose $\sigma_p > 0$ to be a small number compared with the pairwise distances of D. In Procedure 1, we chose σ_p to be approximately $\frac{1}{8}$ times the smallest nonzero pairwise distance among D. In the case that D consists only of discrete observations, this choice aims to ensure that a perturbed data point stays close to its original location. Precisely, for any point $\boldsymbol{x}_i \in D$, $i = 1,\dots,N$, let its closest nonoverlapping neighbor in D be $\boldsymbol{x}_{i_1}$ and let $\boldsymbol{u} \in \mathbb{R}^{q\times 1}$ denote the normalized vector aligned with $\boldsymbol{x}_{i_1} - \boldsymbol{x}_i$. Regarding $\boldsymbol{x}_i$ as the origin, the projected univariate noise (added to $\boldsymbol{x}_i$) along the direction of $\boldsymbol{u}$ will be a normal random variable with mean $\boldsymbol{u}^T\boldsymbol{0}_{q\times 1} = 0$ and variance $\boldsymbol{u}^T(\sigma_p^2\boldsymbol{1}_{q\times q})\boldsymbol{u} = \sigma_p^2$. By choosing σ_p at least as large as $0.125\|\boldsymbol{x}_{i_1} - \boldsymbol{x}_i\|$, we ensure that the probability of the projected univariate noise in the subspace of $\boldsymbol{u}$ being located closer to $\boldsymbol{x}_i$ than $\boldsymbol{x}_{i_1}$ is not smaller than $1 - 6.4 \times 10^{-5}$. In the case that D consists only of continuous observations, the added noise will be negligible. Adding Gaussian noise is actually consistent with the notion of Gaussian kernel density estimation (KDE), which can be viewed as convolving the empirical distribution of D with a Gaussian "noise" density with standard deviation related to the kernel bandwidth.

The third issue is related to the second and regards estimating the density $f(\boldsymbol{x}_i)$. Let $\tilde{\boldsymbol{x}}_i$ denote the perturbed version of $\boldsymbol{x}_i \in D$, $i = 1,\dots,N$. We can consider estimating $f(\boldsymbol{x}_i)$ by $\hat{f}(\boldsymbol{x}_i)$ or by $\hat{f}(\tilde{\boldsymbol{x}}_i)$. When the data distribution is continuous, these two options yield similar results because, by design, the noises added to continuous data are negligible. When the data distribution is discrete, the former choice may seem intuitively preferable because it ensures that whenever $\boldsymbol{x}_i = \boldsymbol{x}_j$, we will have $\hat{f}(\boldsymbol{x}_i) = \hat{f}(\boldsymbol{x}_j)$, $\{i,j\} \in \{1,\dots,N\}$. However, in practice, we observed that our DS algorithm appears to tolerate the standard deviation in the estimated density better than it tolerates bias, and the latter choice provided more diverse subsamples (see Appendix A for an numerical example).

Besides the SIR algorithm (Rubin 1987, 1988), one can also consider adapting the importance support points (ISP) resampling algorithm (algorithm 3 in Huang et al. 2022) into a diversity subsampling algorithm. The ISP resampling algorithm was developed and used for the standard Markov China Monte Carlo setting of approximating a posterior distribution. We choose to adapt SIR in this paper due to computational and storage issues with the ISP resampling algorithm.

Procedure 1 (DS Algorithm: Diversity Subsampling Without Replacement)

Input: n, $D = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N\} \subset [0,1]^q$

1: $N \leftarrow |D|$
2: Standardize D into $[0,1]^q$, and store this new data set as $\tilde{D} = \{\tilde{\boldsymbol{x}}_i\}_{i=1}^N$
3: $n_s \leftarrow \min\{2000, \lfloor \frac{N}{4} \rfloor\}$
4: Randomly select $\{r_1,\dots,r_{n_s}\} \subset \{1,\dots,N\}$ with equal probabilities
5: $D_{\text{sub}} \leftarrow$ all unique points in $\{\tilde{\boldsymbol{x}}_{r_j}\}_{j=1}^{n_s}$
6: Repeat lines 4 and 5 until $|D_{\text{sub}}| > 1$
7: $\sigma_p \leftarrow \frac{1}{8}\min_{\{\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}}\}\subset D_{\text{sub}}}\|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}\|_2$
8: **for** $i = 1,\dots,N$ **do**
9: Independently generate $\boldsymbol{Z}_i \sim \mathcal{N}(\boldsymbol{0}_q, \boldsymbol{1}_{q\times q})$
10: $\tilde{\boldsymbol{x}}_i \leftarrow \tilde{\boldsymbol{x}}_i + \sigma_p \boldsymbol{Z}_i$
11: **end for**
12: Estimate density $\hat{f}(\tilde{\boldsymbol{x}}_i)$, for each $i = 1,2,\dots,N$ (see Appendix B)
13: $\hat{f}(\boldsymbol{x}_i) \leftarrow \hat{f}(\tilde{\boldsymbol{x}}_i)$, for each $i = 1,2,\dots,N$
14: Set index array $I_1 \leftarrow \{1,\dots,N\}$
15: Select j_1 randomly from I_1 with probabilities given by Equation (1)
16: $\mathcal{S}_1 \leftarrow \{\boldsymbol{x}_{j_1}\}$
17: Set an estimated number of density estimation updates U (e.g., $U \leftarrow 10$)
18: Set the number of points to select before conducting a density update. $M \leftarrow \max\{100, \lfloor \frac{n}{U} \rfloor\}$
19: Set *UpdateState* $\leftarrow \{M, 2M, \dots, \lfloor \frac{n}{M} \rfloor M\}$
20: **for** $k = 2,\dots,n$ **do**

21: $I_k \leftarrow I_{k-1} \setminus j_{k-1}$
22: **if** $k-1 \in$ *UpdateState* **then**
23: Re-Estimate $\{\hat{f}(\tilde{\boldsymbol{x}}_i)\}_{i\in I_k}$ by efficient updating (See Appendix B for details)
24: Reset $\hat{f}(\boldsymbol{x}_i) \leftarrow \hat{f}(\tilde{\boldsymbol{x}}_i)$, for each $i \in I_k$
25: **end if**
26: Select j_k randomly from I_k with probabilities given by Equation (2)
27: $\mathcal{S}_k \leftarrow \mathcal{S}_{k-1} \cup \{\boldsymbol{x}_{j_k}\}$
28: **end for**
Output: Subsample set $\mathcal{S}_n = \{\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \ldots, \boldsymbol{x}_{j_n}\}$

2.1. Sampling-with-Replacement Version of the DS Algorithm

In certain situations, subsampling from D with replacement may be relevant. In our primary setting where D is unlabeled data, and the intent is to observe a response Y for each point in $\mathcal{S}_n$, sampling with replacement is only relevant if it is possible to experiment on (i.e., observe a response y_i for) the same subject $\boldsymbol{x}_i$ multiple times, each time observing a potentially different stochastic value for y_i. As an example, suppose $\boldsymbol{x}_i$ is a set of measured characteristics of sample i of a chemical compound from a large set D of samples over which $\boldsymbol{X}$ varies randomly, y_i is some output property of a subsequent chemical reaction with reactants taken from sample i, and y_i can vary randomly if the reaction is repeated using reactants from sample i again. In this case, one may want to use sampling from D with replacement, so the same sample can be experimented on multiple times if needed. Conversely, suppose $\boldsymbol{x}_i$ is a set of clinical, phenotypical, and demographic characteristics of a patient i having a particular condition, and y_i is the efficacy of a treatment for that condition administered to patient i. In this case, it is not meaningful to apply an experimental treatment to the same patient twice, so sampling with replacement becomes irrelevant. See Rubin (1988) for additional discussion of sampling with or without replacement.

For the sampling-with-replacement version of the DS algorithm, denoted by DS-WR, at each iteration $k \in \{1, 2, \ldots, n\}$, we select the next point $\boldsymbol{x}_{j_k}$ from D as follows. The index $j_k \in \{1, 2, \ldots, N\}$ is selected with probability

$$P(j_k | j_1, \ldots, j_{k-1}) = P(j_k) = \frac{\frac{1}{\hat{f}(\boldsymbol{x}_{j_k})}}{\sum_{j=1}^{N} \frac{1}{\hat{f}(\boldsymbol{x}_j)}}, \tag{3}$$

independently of which points have been selected in earlier iterations. The DS-WR algorithm is identical to Procedure 1, except that for DS-WR, one repeats line 15 n times to produce the indices $\{j_1, j_2, \ldots, j_n\}$ of the desired subsample of size n, and all lines after line 15 are omitted.

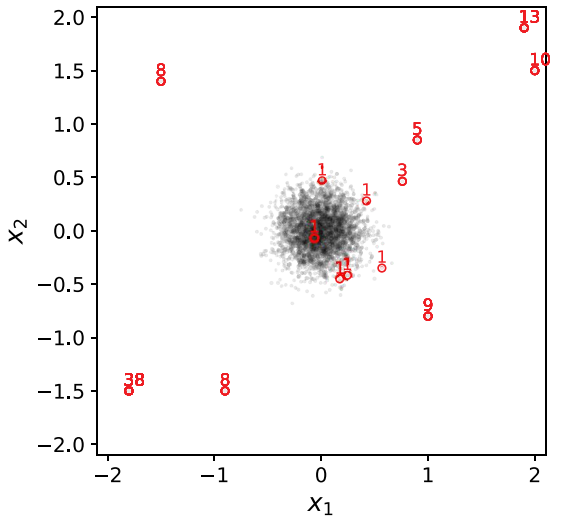
If the DS-WR version is relevant for a particular problem, there is a potential drawback that should be considered if there are severe outliers in the data, as also noted in section 10.4 in Gelman et al. (1995). Most existing density estimation techniques tend to underestimate the density in extremely sparse tail regions of S, where there are only a few outlier points in D. This causes the estimated density at the outlier $\boldsymbol{x}$ points to have extremely low values and therefore very high probabilities of getting selected in an iteration of the DS algorithm. To provide the most diversity, it is generally desirable to select these points for the subsample, which the DS algorithm does. However, in the DS-WR algorithm, the same outlier points may get selected repeatedly for the subsample, which ends up being counterproductive for the diversity goal.

Figure 2 shows an example to illustrate this. The subsample of size $n = 100$ was selected by the DS-WR algorithm from a data set D with $N = 3{,}000$ and $q = 2$. The open circles are the selected points in $\mathcal{S}_n$, and the numbers beside them indicate how many times each point was selected. There are only 14 unique points in the subsample of size 100, and over half of the 100 points are repeated values of only three $\boldsymbol{x}_i$ points that were repeated 38, 13, and 10 times, respectively. The DS algorithm (without replacement, see Procedure 1) would likely have selected each of the outlier points, but only once each, which seems more desirable in this case.

3. Generalization to Customized Nonuniform Subsamples

In this section, we discuss how to generalize the DS method to select subsamples following desired distributions other than uniform. We also provide an example using the generalized DS method to select a subsample

Figure 2. (Color online) Oversampling of Outliers That Can Occur When Sampling with Replacement Is Used



Notes. The open circles are the subsample $\mathcal{S}_n$ with $n = 100$, and the small dots are the set D of size $N = 3{,}000$. The numbers next to the outlier points indicate how many times that point was selected in $\mathcal{S}_n$.

with desired properties without using a well-defined target distribution. For use in various machine learning contexts, the generalized DS algorithm could be used to incorporate information on the response, if they are available. As noted in Joseph and Mak (2021), for some applications this may be more desirable than using a diverse subsample in the input space. The theoretical performance of the generalized DS algorithm is shown in Section 4.

Consider the same setting as in Section 2, and suppose we want to select an i.i.d. subsample from D following some desired distribution with specified density $g(\boldsymbol{x}) = c\tilde{g}(\boldsymbol{x})$ having support $S_g \subset S$. Here $\tilde{g}(\boldsymbol{x})$ is known but c (the constant necessary for $g(\boldsymbol{x})$ to be a proper density) can be unknown. The generalized DS algorithm differs from the DS algorithm only in that each $\boldsymbol{x}_i$ is selected with probability

$$\frac{\frac{\tilde{g}(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)}}{\sum_{k=1}^{N}\frac{\tilde{g}(\boldsymbol{x}_k)}{f(\boldsymbol{x}_k)}}$$

for the generalized DS. Specifically, at iteration $k=1$, the generalized DS algorithm selects the index $j_1 \in \{1,\ldots,N\}$ of the first point $\boldsymbol{x}_{j_1}$, with probability

$$P(j_1) = \frac{\frac{\tilde{g}(\boldsymbol{x}_{j_1})}{f(\boldsymbol{x}_{j_1})}}{\sum_{i=1}^{N}\frac{\tilde{g}(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)}}. \tag{4}$$

At iteration $k \geq 2$, the generalized DS algorithm selects the index $j_k \in \{1,\ldots,N\} \setminus \{j_1,\ldots,j_{k-1}\}$ of the next sampled point $\boldsymbol{x}_{j_k}$ with probability

$$P(j_k|j_1,\ldots,j_{k-1}) = \frac{\frac{\tilde{g}(\boldsymbol{x}_{j_k})}{f(\boldsymbol{x}_{j_k})}}{\sum_{i=1}^{N}\frac{\tilde{g}(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)} - \sum_{i=1}^{k-1}\frac{\tilde{g}(\boldsymbol{x}_{j_i})}{f(\boldsymbol{x}_{j_i})}}. \tag{5}$$

Consequently, Procedure 1 applies to the generalized DS algorithm with only line 15 and line 26 modified according to Equation (4) and Equation (5), respectively. Because N is finite in practice, the performance of the generalized DS algorithm will degrade if the tail of $f(\boldsymbol{x})$ happens to be the mode of $g(\boldsymbol{x})$, because in D there may not be enough points to choose to form a subsample with density $g(\boldsymbol{x})$. The same issue is of course present in the DS algorithm, because the uniform target density also will be much larger than $f(\boldsymbol{x})$ in its tail regions. Thus, the points in D falling in the tail regions of $f(\boldsymbol{x})$ will be depleted too quickly to allow for a uniform subsample over the tails. We consider this phenomenon in our examples in Section 5, where we also observe that this is a unavoidable problem for any existing diversity subsampling algorithm.

We now provide an example using the generalized DS algorithm in the case that $g(\boldsymbol{x})$ is not a well-defined p.d.f. Balancing the diversity property in the input space and some criterion in the response space has been found beneficial for some applications in the literature (Joseph and Mak 2021, Ren et al. 2021). For regression problems, Joseph and Mak (2021) observed that balancing diversity in the input space and the loss function in the response space makes the selected subsample more robust with respect to different modeling strategies. For classification problems in active learning, uncertainty and diversity are two desirable properties of the selected subsample. Many works have found that a subsample incorporating both properties can be more beneficial (Ren et al. 2021). Here we provide a simple classification example illustrating how to use the generalized DS for such purposes.

Consider the following binary classification problem with $q = 2$. The data set D of predictor vectors is the same as in the MGM example described in Section 5.2, and we generate the binary response variable Y (= 0 or 1) via

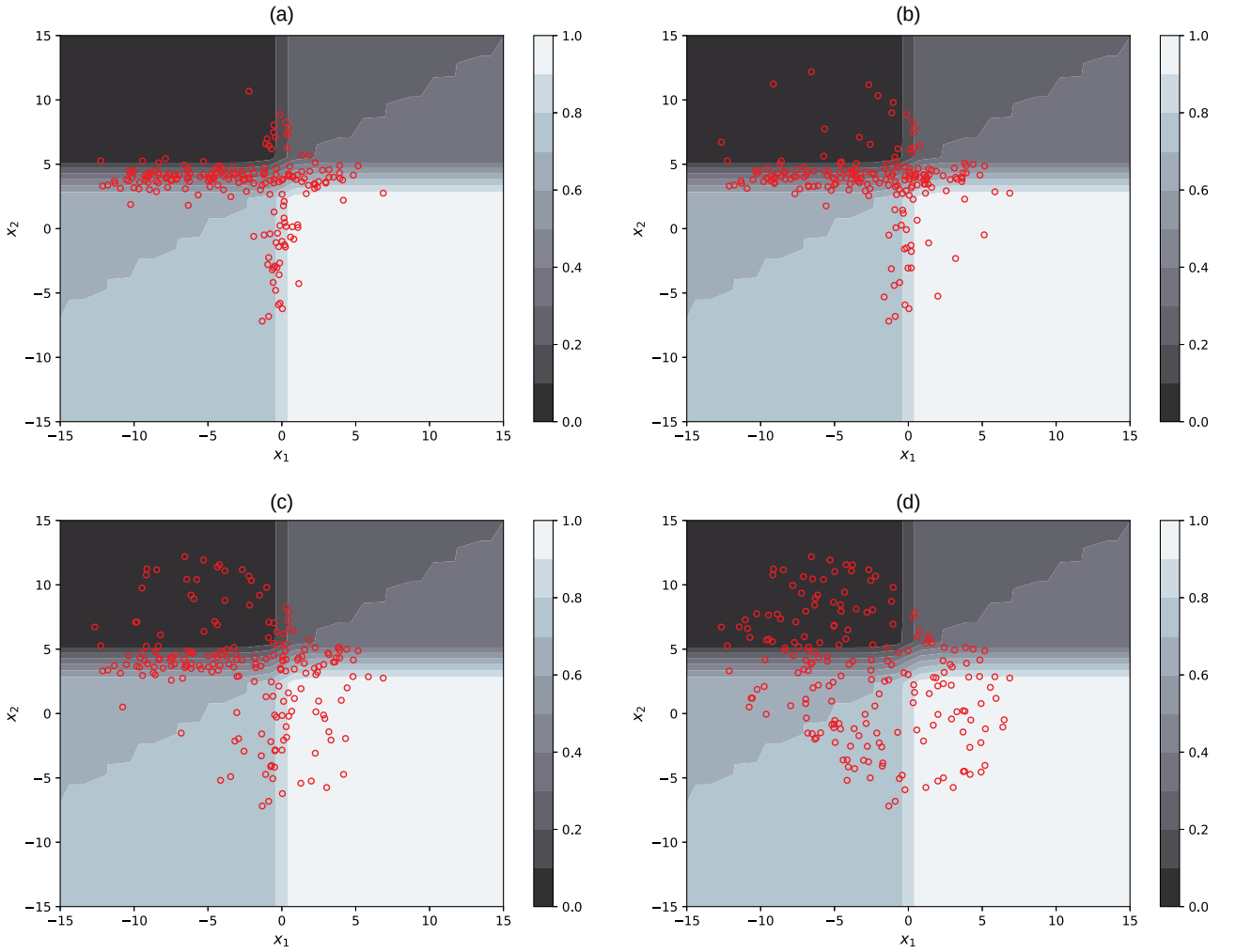
$$\begin{aligned} P(Y=1|\boldsymbol{X}) &= P(Y=1|X_1,X_2) \\ &= 0.3\frac{1}{1+e^{-2X_1}} + 0.7\frac{1}{1+e^{3(X_2-4)}}. \end{aligned} \tag{6}$$

We use the Euclidean norm of the gradient of Equation (6) as a simple surrogate measure of classification uncertainty (a large gradient means the predicted probability $P(Y=1|\boldsymbol{X})$ is changing rapidly in that region, which usually translates to higher classification uncertainty) and denote it as $u(\boldsymbol{x}) \overset{\text{def}}{=} \|\nabla_X P(Y=1|\boldsymbol{X})\|_2$. To incorporate both uncertainty and diversity into the subsample, we let $g(\boldsymbol{x}) = u(\boldsymbol{x}) + u_\alpha$, where u_α is a constant (and thus represents a uniform distribution to promote the diversity, whereas $u(\boldsymbol{x})$ considers the classification uncertainty) that we take to be the lower α quantile of set $\{u(\boldsymbol{x}_i)\}_{i=1}^{N}$ for some specified α. Here $g(\boldsymbol{x})$ typically will not be a well-defined p.d.f. For the generalized DS algorithm, the sampling probability of each data point will be proportional to $g(\boldsymbol{x}_i)/f(\boldsymbol{x}_i)$. Selecting a larger α will promote more diversity and selecting a smaller α will result in more subsample points chosen in the areas with higher uncertainty. Figure 3 shows the selected subsamples for various α. The colorbar shows the values of Equation (6). We observe that when $\alpha = 0$, most of the selected points locate close to the decision boundary $\boldsymbol{x} : P(Y=1|\boldsymbol{x}) = 0.5$ (Figure 3(a)), where the highest uncertainty occurs in this model. Using a larger α, such as $\alpha = 50\%$ allows more subsample points in other regions in the predictor space (Figure 3(c)). When $\alpha = 100\%$, the subsample appears to be quite diverse.

4. Theoretical Performance Analysis

This section presents results on the convergence rate of the generalized DS algorithm (Section 3) when the true $f(\boldsymbol{x})$ is known (Theorem 1). For convenience, we say that a function $u(N)$ of N is $O(N^r)$ if $\exists c > 0, k > 0$, (c, k are both constants that do not depend on N), such that $|u(N)| \leq cN^r$, $\forall N \geq k$. We say $u(N)$ is $o(N^r)$ if $\forall c > 0, \exists k(c) > 0$ that does not depend on N, such that $|u(N)| < cN^r$, $\forall N \geq k(c)$.

Figure 3. (Color online) Subsample Selected by Generalized DS Method for the Binary Classification Example



Notes. The open circles indicate selected subsample points. The color bar on the right side of each plot shows the value of the event probability specified in Equation (6). (a) $\alpha = 0\%$. (b) $\alpha = 25\%$. (c) $\alpha = 50\%$. (d) $\alpha = 100\%$.

When the true p.d.f of the data distribution is available, the generalized DS algorithm (Section 3) is the same as the SIR algorithm (Rubin 1987, 1988), for which Skare et al. (2003) proves convergence and which we restate as Theorem 1.

Theorem 1 (Theorem 2, Part 1 in Skare et al. 2003). *Let* $\boldsymbol{X}^N = (\boldsymbol{X}_i : i = 1,2,\ldots,N)$, *where* $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N \in \mathbb{R}^q$ *are i.i.d. copies of random vector* $\boldsymbol{X}$ *having density function* $f(\boldsymbol{x})$ *with support* $S \subset \mathbb{R}^q$. *Let* $g(\boldsymbol{x}) = c\tilde{g}(\boldsymbol{x})$ *be the desired density function for the selected subsample that is known up to a constant* $c \in \mathbb{R}^+$. *Assume that*

A.1. *The support of* $g(\boldsymbol{x})$, $S_g \subset S$

A.2. *The variance of* $\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})}$, $Var\left[\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})}\right] = \int_S \frac{g^2(\boldsymbol{x})}{f(\boldsymbol{x})} d\boldsymbol{x} - 1 = \sigma^2 < +\infty$

Suppose $n \geq 1$ *is fixed. For each* $N = 1,2,\ldots,$ *let* $(\boldsymbol{Z}_N^k : k = 1,2,\ldots,n)$ *be drawn sequentially from* $\boldsymbol{X}^N$, *without replacement, according to the conditional probability mass function (p.m.f.) (for* $k = 1$*)*

$$P(\boldsymbol{Z}_N^1 = \boldsymbol{X}_i | \boldsymbol{X}^N) = \frac{\frac{\tilde{g}(\boldsymbol{x})}{f(\boldsymbol{X}_i)}}{\sum_{j=1}^N \frac{\tilde{g}(\boldsymbol{x})}{f(\boldsymbol{X}_j)}}, \quad (i = 1,\ldots,N), \tag{7}$$

and p.m.f. (for $k \geq 2$*)*

$$P(\boldsymbol{Z}_N^k = \boldsymbol{X}_i | \boldsymbol{X}^N, \boldsymbol{Z}_N^1, \ldots, \boldsymbol{Z}_N^{k-1}) = \frac{\frac{\tilde{g}(\boldsymbol{X}_i)}{f(\boldsymbol{X}_i)}}{\sum_{j=1}^N \frac{\tilde{g}(\boldsymbol{X}_j)}{f(\boldsymbol{X}_j)} - \sum_{j=1}^{k-1} \frac{\tilde{g}(\boldsymbol{Z}_N^j)}{f(\boldsymbol{Z}_N^j)}},$$
$$(i \in \{1,2,\ldots,N\} \setminus \{j_1^N, j_2^N, \ldots, j_{k-1}^N), \tag{8}$$

where $j_1^N, j_2^N, \ldots, j_n^N$ *are the indices of the sampled observations in* $\boldsymbol{X}^N$, *that is,* $\boldsymbol{Z}_N^k = \boldsymbol{X}_{j_k^N}$, *for* $k = 1,2,\ldots,n$. *Denote the joint p.d.f. of* $\boldsymbol{Z}_N^1, \ldots, \boldsymbol{Z}_N^n$ *as* $p_N^n(z_1,\ldots,z_n)$. *Then as* $N \to +\infty$,

$$\frac{p_N^n(z_1,\ldots,z_n)}{\prod_{k=1}^n g(z_k)} - 1 = \frac{1}{N}\left(\sum_{k=1}^n k\left(1 - \frac{g(z_k)}{f(z_k)} + \sigma^2\right)\right) + O\left(\frac{1}{N^2}\right) \to 0. \tag{9}$$

Furthermore, when $\sup_{x\in S}\frac{g(x)}{f(x)} < +\infty$, *the convergence of* $p_N^n(z_1,\ldots,z_n)$ *is uniform.*

In practice, $f(x)$ must be estimated, in which case the generalized DS algorithm may not have the convergence properties as SIR, due to inaccuracies in estimating $f(x)$. It is not unreasonable to assume that if $\hat{f}(x)$ converges to $f(x)$ at a sufficient rate, then the sample produced by the DS algorithm using $\hat{f}(x)$ will converge to the target distribution. Although we were unable to prove this under general convergence conditions that are known to hold for our estimator of $f(x)$, preliminary results indicate that convergence of the DS algorithm holds for certain specified conditions on the density estimator. More importantly, our numerical results presented later demonstrate that the DS algorithm with estimated $f(x)$ does indeed produce samples that effectively approximate the target distribution.

Remark 1. Theorem 1 apply to the DS algorithm (Procedure 1) for producing a uniform sample by setting $\tilde{g}(x) = 1,\ \forall x \in S_g$.

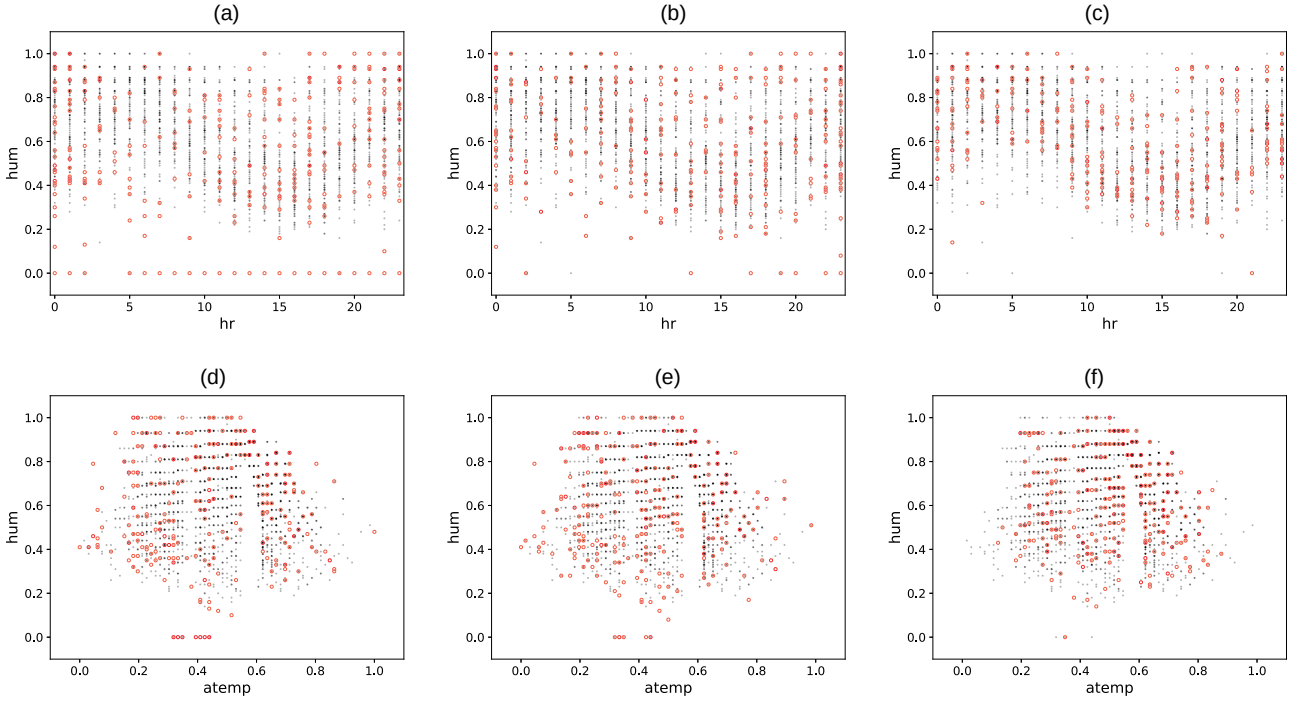
5. Numerical Performance Analysis

In this section, we test the performance of the DS algorithm using data sets following various distributions and compare it with competing subsampling methods. Section 5.1 applies various subsampling algorithms to a real-world data set and visually compares the diversity of the selected subsamples. Section 5.2 discusses numerical performance evaluation criteria and the experimental settings for the comparative examples. Section 5.3 provides numerical results comparing uniformity performances of DS with competing algorithms, and Section 5.4 compares runtime of the algorithms.

5.1. Example: Bike-Sharing Data

We apply DS, scSampler-sp1 (Song et al. 2022b), and random sampling to the open-source bike-sharing data set used in Fanaee-T and Gama (2014). The data set is of size 17,379, and we included the following 6 variables in our experiments: hr, holiday, weathersit, atemp, hum, and windspeed. Here hr records at which hour of a day the number of rented bikes was counted; holiday indicates whether the day is a holiday; weathersit uses four grades to record the weather situation; atemp stands for apparent temperature; hum records humidity and windspeed stores the wind speed. A subsample of size $n = 300$ was selected from the data set using each method (we standardized the data into a unit hypercube for DS and scSampler-sp1), and Figure 4 show projected two-dimensional (2D) scatter plots in the hum-hr and hum-atemp subspaces. We observe from Figure 4 that the subsamples selected by DS and scSampler-sp1 appear more spread-out and diverse than the one selected by

Figure 4. (Color online) Projected 2D Scatter Plots for the Selected Subsamples of Size $n = 300$ from the Bike-Sharing Data Set (Fanaee-T and Gama 2014) for Three Different Methods (DS, scSampler-sp1 (Song et al. 2022b), and Random Sampling)



Notes. The small dots represent a size 2,000 random subsample from the data, and the open circles represent the selected subsample points. (a) DS. (b) scSampler-sp1. (c) Random sampling. (d) DS. (e) scSampler-sp1. (f) Random sampling.

random sampling. For example, both DS and scSampler-sp1 selected points at the lowest hum level, whereas random sampling did not. Although for brevity we omit projected scatter plots for other pairs of variables, similar conclusions apply for them.

5.2. Experimental Settings

As a performance evaluation criterion, we use the sample version of energy distance (Székely 2003, Székely and Rizzo 2004) to measure the extent to which a subsample is drawn from an i.i.d. uniform distribution over some specified $\Omega \subset S$, where Ω is compact. We provide further details of the choice of Ω below. Let $\boldsymbol{X}, \boldsymbol{U}$ be mutually independent random variables in $\mathbb{R}^q$. The sample version of energy distance is as follows. Let $\mathcal{A}_n = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$ and $\mathcal{U} = \{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_{N_1}\}$ be i.i.d. samples of size n and N_1, respectively, from the same distributions followed by $\boldsymbol{X}$ and $\boldsymbol{U}$, respectively. The energy distance between samples $\mathcal{A}_n$ and $\mathcal{U}$ is (Székely 2003)

$$e(\mathcal{A}_n,\mathcal{U}) = \frac{2}{nN_1}\sum_{i=1}^{n}\sum_{j=1}^{N_1}\|\boldsymbol{X}_i - \boldsymbol{U}_j\| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\boldsymbol{X}_i - \boldsymbol{X}_j\| - \frac{1}{N_1^2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_1}\|\boldsymbol{U}_i - \boldsymbol{U}_j\|. \tag{10}$$

The Euclidean norm will be used throughout this paper unless otherwise specified. By proposition 1 in Székely (2003), as $n \to \infty$ and $N_1 \to \infty$, one can show that $E[e(\mathcal{A}_n,\mathcal{U})] \to 0+$ if and only if $\boldsymbol{X} \overset{\mathrm{D}}{=} \boldsymbol{U}$. Here $\overset{\mathrm{D}}{=}$ denotes equality in distribution.

We use Equation (10) in the following way. We choose $\mathcal{U}$ to be a large uniform sample over Ω of size $N_1 >> n$, and we take $\mathcal{A}_n$ to be the subsample of size n selected from D using some subsampling algorithm. If points in subsample $\mathcal{A}_n$ (subsample points located outside of Ω are excluded under this criterion) follow an i.i.d. uniform distribution over Ω, $E[e(\mathcal{A}_n,\mathcal{U})]$ should be close to zero if n is sufficiently large (we take a fixed large N_1 in this paper), and the more $\mathcal{A}_n$ differs from an i.i.d. uniform distribution over Ω, the larger we expect $E[e(\mathcal{A}_n,\mathcal{U})]$ to be. For selected experiments, we also report the p values for testing the null hypothesis that $\mathcal{A}_n$ and $\mathcal{U}$ are generated from the same distribution using the test proposed in Székely and Rizzo (2004).

Intuitively, to produce a diverse and space-filling subsample, it is typically desirable to select all or most of the points in the regions for which $f(\boldsymbol{x})$ has low-density, which we denote by Ω^c, where $S = \Omega \cup \Omega^c$. Because we cannot expect uniformity over both Ω and Ω^c for large n (unless N is extremely large) as the points in Ω^c will be quickly depleted, we use a separate measure for performances within Ω^c. Because data are sparse in Ω^c, it is generally desirable to select as high a percentage of points in Ω^c as possible. Consequently, we define the following low-density region sampling ratio to measure this property. Suppose Ω is chosen such that $\forall \boldsymbol{x} \in \Omega, f(\boldsymbol{x}) \geq \delta$ and $\forall \boldsymbol{x} \in \Omega^c, f(\boldsymbol{x}) < \delta$, for some specified value δ. Then for a subsample $\mathcal{S}_n$ of size n selected from D, the low-density region sampling ratio of $\mathcal{S}_n$ is defined as

$$r(\mathcal{S}_n) = \frac{\sum_{k=1}^{n} \boldsymbol{I}_{\{\boldsymbol{x}_{j_k} \in \Omega^c\}}\left(1 - \boldsymbol{I}_{\{j_k \in \{j_l\}_{l=1}^{k-1}\}}\right)}{\sum_{i=1}^{N} \boldsymbol{I}_{\{\boldsymbol{x}_i \in \Omega^c\}}}, \tag{11}$$

where $\boldsymbol{I}_{\{\cdot\}}$ denotes the indicator function. For instance, $\boldsymbol{I}_{\{\boldsymbol{x}_{j_k} \in \Omega^c\}}$ equals one if $\boldsymbol{x}_{j_k} \in \Omega^c$, and equals zero otherwise. Only unique subsample indices are used in Equation (11). In situations where a uniform subsample over S of size n will exhaust points in Ω^c, a larger $r(\mathcal{S}_n)$ is desirable, with $r(\mathcal{S}_n) = 1$ typically being the ideal goal (in conjunction with a uniform distribution over Ω).

For the examples, we consider data sets in which each element of $\boldsymbol{X}$ follows independent Gaussian, gamma, exponential, geometric, and multivariate Gaussian mixture (MGM) distributions. We refer to these five different data distributions as the normal, gamma, exponential, geometric, and MGM. The normal example illustrates the performances of the subsampling algorithms when the data distribution is symmetric; the mode of the density is achieved in the interior region of S; S is unbounded. The gamma example illustrates the performances when the data distribution is skewed and S had boundaries but is unbounded. The exponential distribution, which is an extreme case of the gamma distribution, illustrates the performances when the data distribution is extremely skewed and the mode is located at the boundary. The MGM examples are to compare the subsampling algorithms when the data distribution is multimodal with correlated predictors. We also include geometric examples to illustrate the performances of different algorithms when the data distribution is discrete. For all examples in this section, the size of D is $N = 10^4$ for $q = 2$ and $N = 10^5$ for $q = 10$.

The data distributions of each simulation example are as follows. For the normal, gamma, and exponential examples, the elements of $\boldsymbol{X}$ are independent and the data distribution has density $f(\boldsymbol{x}) = \prod_{j=1}^{q} f(x_j)$, where $f(x) = e^{-x^2/2}/\sqrt{2\pi}$, $f(x) = xe^{-x/2}/(4\Gamma(2))$ and $f(x) = e^{-x}$, respectively, where $\Gamma()$ denotes the gamma function. For the MGM example, D was generated as a mixture of two Gaussian clusters in $\mathbb{R}^q$. The true density function is as follows:

$$f(\boldsymbol{x}) = 0.5 f_1(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + 0.5 f_2(\boldsymbol{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}), \tag{12}$$

where

$$f_i(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)\right), \tag{13}$$

for $i = 1, 2$. That is, the two Gaussian clusters have a

common covariance matrix $\boldsymbol{\Sigma}$ but different means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$. We let $\boldsymbol{\mu}_1 = \mathbf{0}_q$ and $\boldsymbol{\mu}_2 = 5.0((-1)^i)_{i=1}^q$. For the common covariance matrix $\boldsymbol{\Sigma}$, we set $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_{q\times q} + \alpha \boldsymbol{a}\boldsymbol{a}^T$. Here $\sigma^2 = 4.0$, $\alpha = 1.0$, $\boldsymbol{I}_{q\times q} \in \mathbb{R}^{q\times q}$ is the identity matrix, and $\boldsymbol{a} = (a_i)_{i=1}^q$, for $a_i = 0.2(i-2)(-1)^i$. The data distribution of the geometric example also has independent covariates; thus the joint p.m.f. is $f(\boldsymbol{x}) = \prod_{j=1}^q f(x_j)$, where $f(x) = (1-p)^{x-1}p$. We set $p = 0.5$ for $q = 2$ and $p = 0.9$ for $q = 10$. The supports of the p.d.f. functions of the normal, gamma, exponential, MGM, and geometric examples are, respectively, $S = \mathbb{R}^q, \mathbb{R}_+^q, \mathbb{R}_+^q, \mathbb{R}^q, \mathbb{Z}_+^q$.

We choose $\Omega = \{\boldsymbol{x} \in S | f(\boldsymbol{x}) \geq \delta\}$, where δ is chosen such that 99% of the data points in D fall inside of Ω when $q = 2$ and 99.9% of the data points in D do so when $q = 10$. Figure 5 depicts Ω for selected examples with $q = 2$. To be clear, for discrete distributions, the target uniform distribution is a discrete uniform distribution over S. This distinction between continuous and discrete distributions is not used in any way in the DS algorithm, although it is used in our performance measures.

5.3. Numerical Results for Performance Comparisons

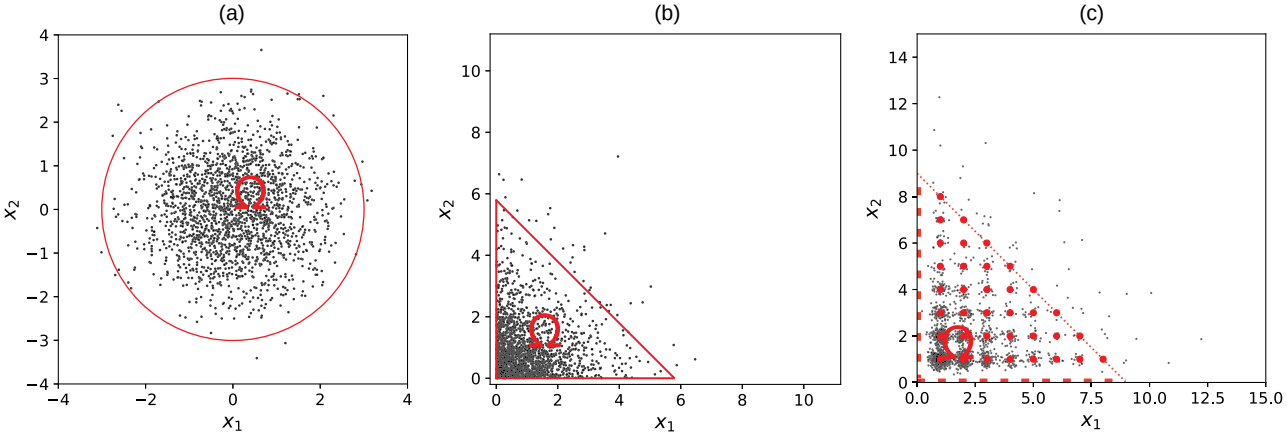
The competing algorithms that we compare include scSampler-sp1, scSampler-sp16, DPP, WSE, (weighted sample elimination algorithm, a heuristic for PDS proposed by Yuksel 2015), and simple random sampling. Here scSampler-sp1 (Song et al. 2022b) serves as an efficient heuristic for the CADEX algorithm. scSampler-sp16 (Song et al. 2022b) is included as a modification of scSampler-sp1 for better computational efficiency. We used published codes for scSampler-sp1 and scSampler-sp16 (Song et al. 2022a), DPP (Biyik 2019), and WSE (Yuksel 2016). We omit the results for RD ALR in this paper, as we found that it generally did not perform as well as other methods.

The implementation details for competing methods are as follows. For the two versions of the scSampler algorithm, no additional parameters need to be set. For DPP, we set hyperparameters $\alpha = 4.0$, $\gamma = 0.0$ and *steps* = 0 (Biyik et al. (2019) recommended setting $\alpha \geq 2.0$ and $\gamma = 0.0$ in their paper to get the most diverse subsample; we chose *steps* = 0 for computational efficiency at the cost of the resulted subsample being deterministic). For WSE, we used the default weighting function in Yuksel (2016) and set *Progressive* = *True* to make the selected subsample sequential.

Figure 6 plots the average value of $e(\mathcal{S}_n, \mathcal{U})$ across 200 replicates of the experiments (its standard error across the 200 replicates was negligibly small and is omitted from the figures). For DPP and WSE, only a single replicate was used, since WSE and DPP (with steps = 0) are deterministic and produce the same subsamples each replicate. The results for the low density region sampling ratio (Equation (11)) are consistent with the $e(\mathcal{S}_n, \mathcal{U})$ results and are shown in Appendix D. The size of the large uniform sample $\mathcal{U}$ inside Ω is the same as $|D|$. As a reference point, we have included the $e(\mathcal{S}_n, \mathcal{U})$ measures for true uniform samples (denoted as uniform sampling) inside Ω.

We assessed the performances of the subsampling algorithms for $n = 20, 520, \ldots, 4020$ at $q = 2$ (DPP was only tested for $n = 20, 250, \ldots 1{,}860$ due to expensive computational cost; see Section 5.4) and for $n = 200, 900, \ldots, 6500$ at $q = 10$.[1] DPP was not included in the comparisons at $q = 10$ as its computation costs was excessive.

Figure 5. (Color online) Scatter Plots of D and the Chosen Regions Ω for the Normal, Exponential, and Geometric Examples Used in This Section for $q = 2$



Notes. For normal and exponential examples, the curve indicates the boundary of Ω. For the geometric example, the dotted lines show region $\{\boldsymbol{x} \in \mathbb{R}^q | f(\boldsymbol{x}) \geq \delta\}$, and the union of solid large dots corresponds to the region $\Omega = S \cap \{\boldsymbol{x} \in \mathbb{R}^q | f(\boldsymbol{x}) \geq \delta\}$, which consists of a finite number of discrete points. Two thousand randomly chosen data points are plotted in this graph for each example and are shown by tiny dots. The geometric data are perturbed for easy visualization. (a) Normal example. (b) Exponential example. (c) Geometric example.

Figure 6. (Color online) Averaged $e(\mathcal{S}_n,\mathcal{U})$, as a Function of n, for Subsamples Selected by DS, Random Sampling from D (Random Sampling), scSampler-sp1, scSampler-sp16, WSE, and DPP

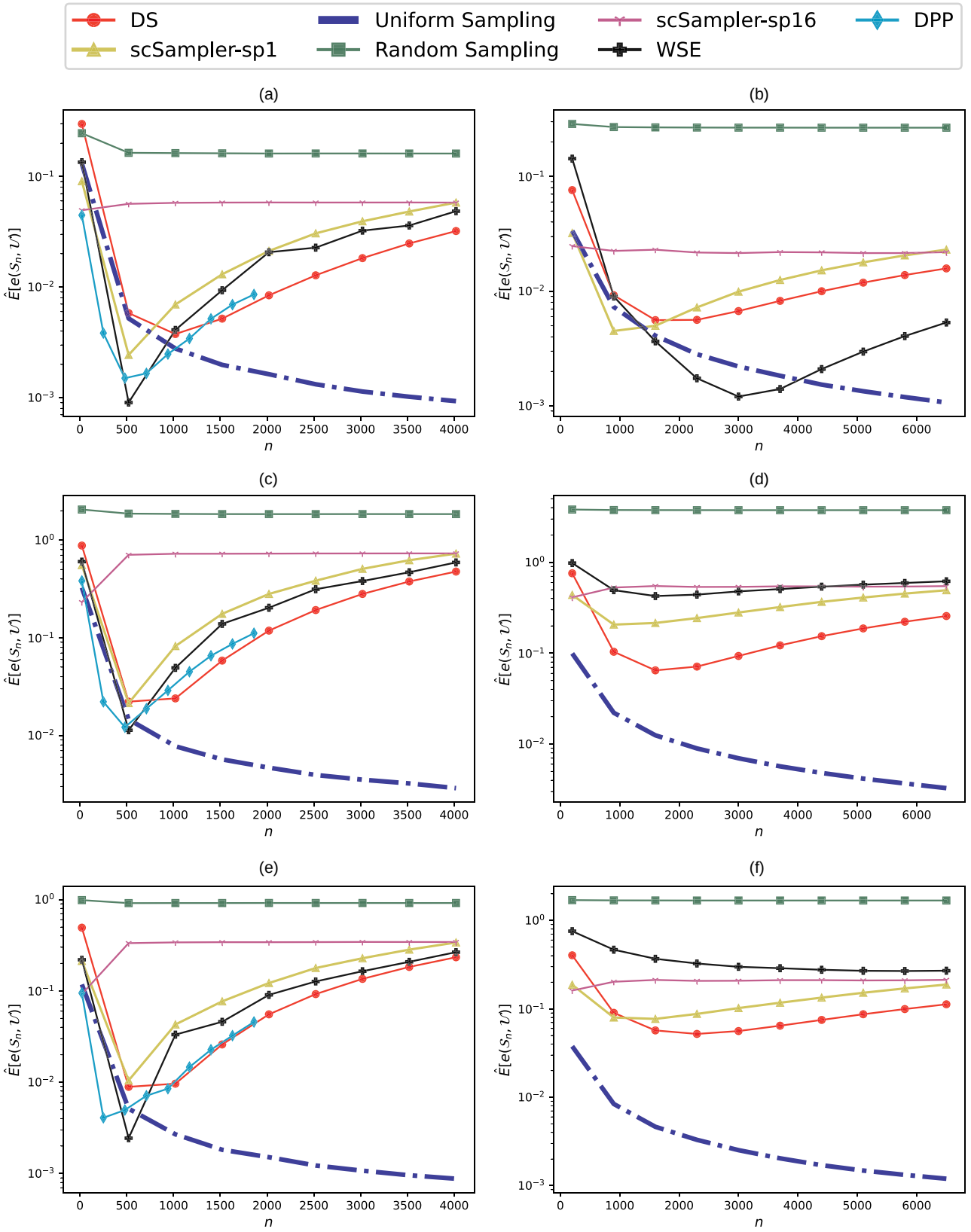
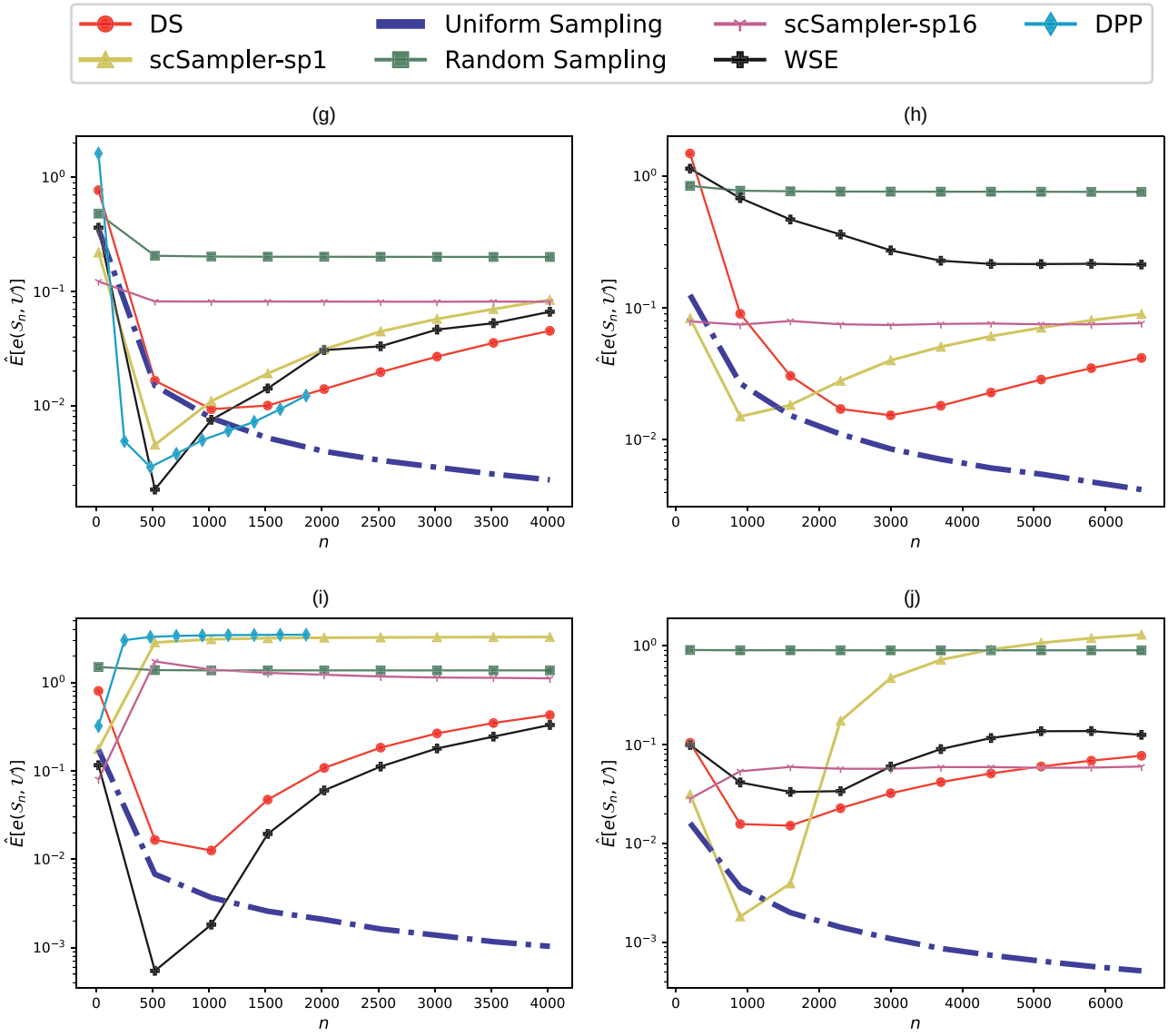


Figure 6. (Continued)



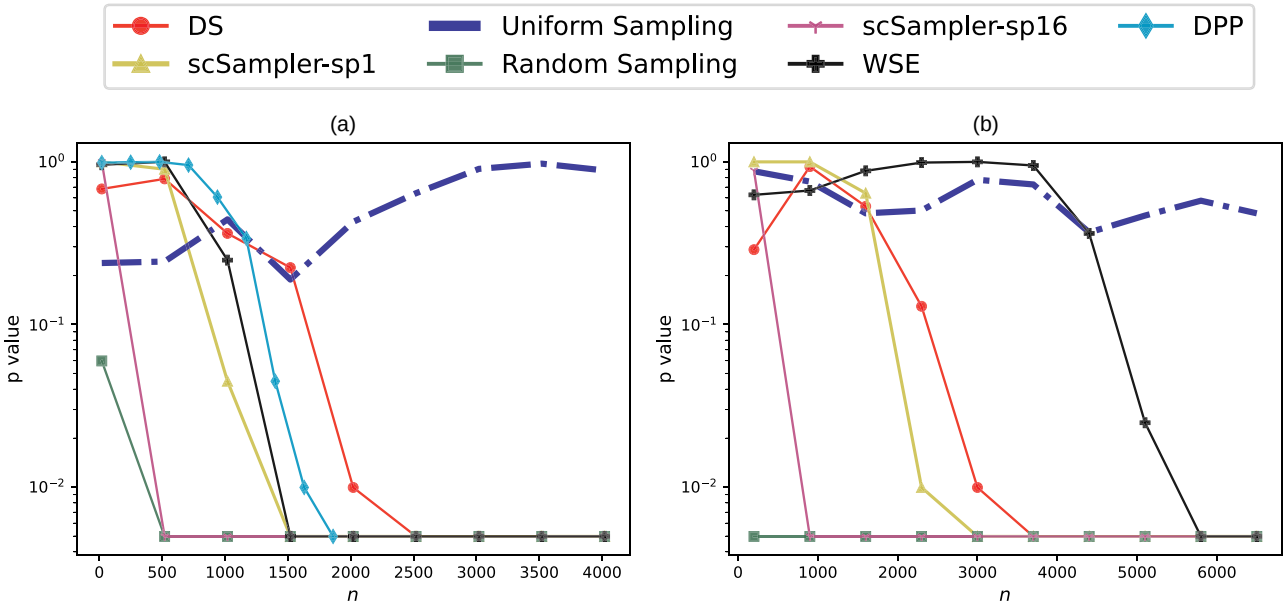
Notes. Uniform Sampling serves as a reference (the closer the performance of an algorithm is to the Uniform Sampling curve, the better). (a) Normal, 2D. (b) Normal, 10D. (c) Gamma, 2D. (d) Gamma, 10D. (e) Exp, 2D. (f) Exp, 10D. (g) MGM, 2D. (h) MGM, 10D. (i) Geometric, 2D. (j) Geometric, 10D.

Figure 6 shows that average value of $e(\mathcal{S}_n,\mathcal{U})$ for uniform sampling converges to zero as n increases, whereas for most subsampling algorithms it first decreases and then increases instead of monotonically decreasing with n. This is because uniform sampling generates sample points from Ω directly, whereas the subsampling algorithms select points from a finite D without replacement. After points in D in the lower-density regions in Ω are depleted, the subsampling algorithms can no longer produce uniform samples. We discuss this phenomenon in more detail in Appendix C.

We also observe from Figure 6 that some algorithms have an average $e(\mathcal{S}_n,\mathcal{U})$ measure that is actually less than uniform sampling, such as DPP and WSE for the normal example at $q=2$ and $n \le 1000$ (Figure 6(a)). This is a well-known phenomenon, by which points on a uniform grid can have a lower $e(\mathcal{S}_n,\mathcal{U})$ measure than a truly uniform sample (Mak and Joseph 2018). For instance, in separate experiments (not shown here, for brevity), we found that, a grid of size $n=100$ in $[0,1]^2$ often has a smaller sample energy distance to $\mathcal{U}$ than a true uniformly distributed sample. Consequently, we can view a subsampling algorithm whose performance is as close to the uniform sampling as possible as the most effective method in terms of selecting i.i.d. uniformly distributed subsamples over Ω. From Figure 6, we see that overall, the average sample energy distances of DS deviates least from the uniform sampling results for most cases compared with other subsampling algorithms.

For the normal examples in 2D and 10D, we also test whether the subsamples selected by various

Figure 7. (Color online) p Values for Equal Distribution Tests (Székely and Rizzo 2004) Between $\mathcal{S}_n$ and $\mathcal{U}$, as a Function of n, for Various Methods and for the Normal Examples in 2D and 10D



Notes. Uniform Sampling serves only as a benchmark (the closer the performance of an algorithm to the Uniform Sampling curve, the better). (a) Normal, 2D. (b) Normal, 10D.

algorithm are equal in distribution to $\mathcal{U}$ using the equal-distribution test proposed by Székely and Rizzo (2004) and implemented as the eqdist.etest function in the R package energy (Rizzo and Szekely 2022). The value of parameter R for the eqdist.etest function was set to be 200 (i.e., the precision of the estimated p value was set to 5×10^{-3}), and the resulting p values are shown by Figure 7. For computational reasons, the size of $\mathcal{U}$ was set to 10,000 for this experiment. Figure 7 appears consistent with Figure 6, (a) and (b): In Figure 7(a), DS had the largest p value for most n, and in Figure 7(b), only WSE had a larger p value for most n. However, as discussed in Section 5.4, WSE has far larger (often prohibitively large) computational expense than DS.

We also repeated the experiments in Figure 6 using data sets in which observations were replicated for the continuous distributions. Specifically, we generated a data set of size $N/5$ and then replicated this data set five times to generate the set D of size N, which we refer to as replicated data sets. The results using the replicated data sets are shown by Figure 8. Compared with Figure 6, the DS performance has barely changed, whereas the scSampler-sp1 performance has degraded significantly. We conclude that the performance of DS relative to the existing methods further improves, often substantially, for replicated data. The results in Figure 8(a) are consistent with the visualization in Figure 1: at $n = 2{,}000$, the performances of scSampler-sp1 are comparable with random sampling, whereas the performances of DS are substantially better.

5.4. Computation Time Comparisons

Table 1 provides runtimes for the DS, scSampler-sp1, scSampler-sp16, DPP, and WSE[2] algorithms. We implemented the DS algorithm in Python 3 and tested the runtimes of the previously mentioned algorithms on the Quest High Performance Computing Cluster of Northwestern University. The runtimes were for the normal examples (runtime is relatively invariant to the underlying distribution) with varying data set sizes. The reported runtime statistics in Table 1 are in seconds and were averaged over 100 independent replications, although there was not much replicate-to-replicate variability. NA indicates that a subsampling algorithm took longer than one hour to run (for each replicate), in which case the experiment was terminated, and the approach was considered to be computationally prohibitive for that case.

From Table 1, DS and scSampler-sp16 are far more efficient than other methods, often multiple orders of magnitude faster. One should keep in mind, however, that the uniformity performance of scSampler-sp16 was overall far inferior to DS in Figures 6 and 8. For $n \geq 1{,}000$, the runtime of DS remains relatively unchanged with varying n, whereas the runtime of scSampler-sp16 (and of scSampler-sp1 and DPP) grows super linearly with n increasing. Consequently, for the largest n in Table 1, DS becomes faster than scSampler-sp16. Moreover, for larger scale examples with even larger n, we can expect DS to be substantially faster than even scSampler-sp16 (in addition to having substantially better performance).

Figure 8. (Color online) Averaged $e(\mathcal{S}_n,\mathcal{U})$, as a Function of n, Using Replicated Data, for Subsamples Selected by DS, Random Sampling from D (Random Sampling), scSampler-sp1, scSampler-sp16, WSE, and DPP

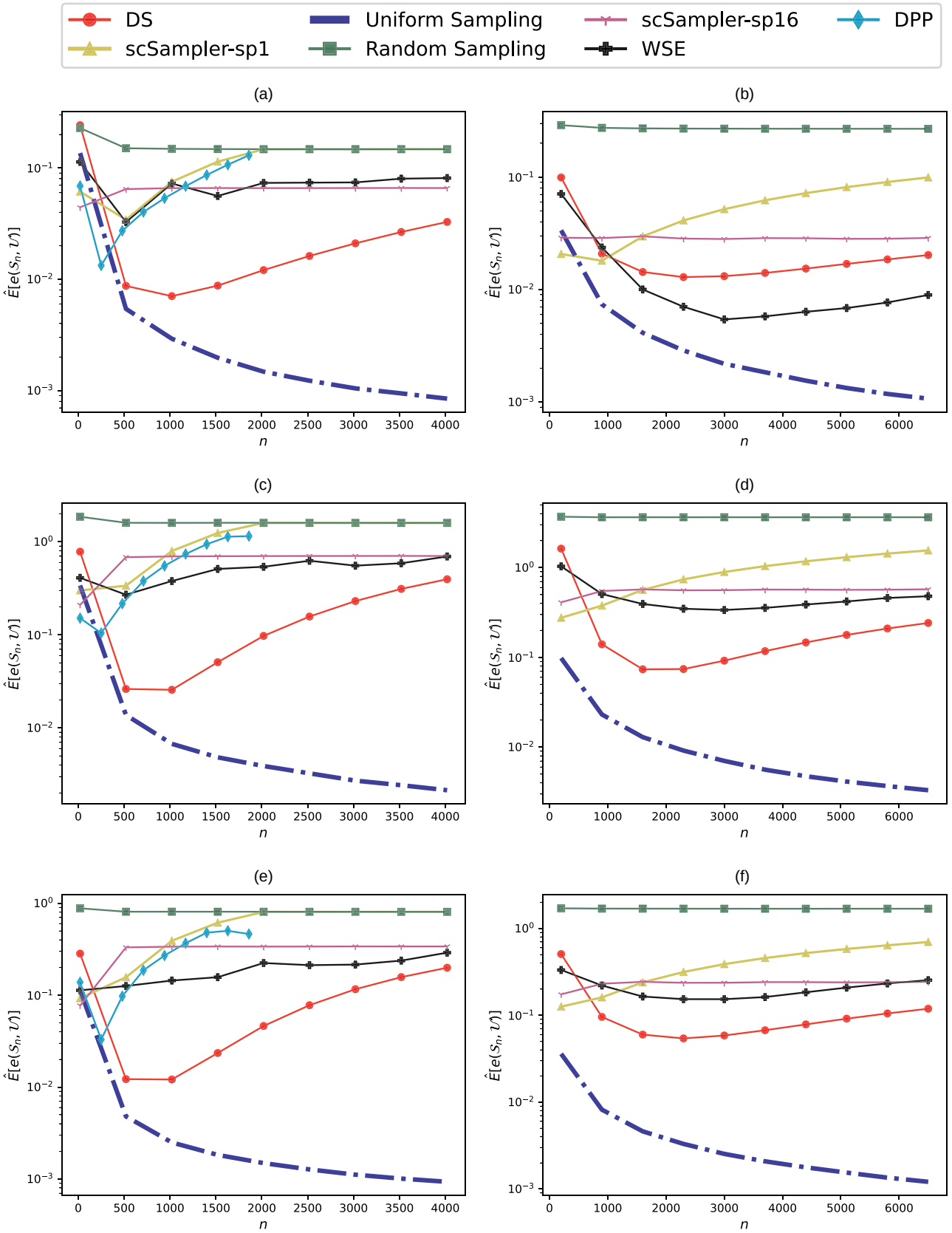
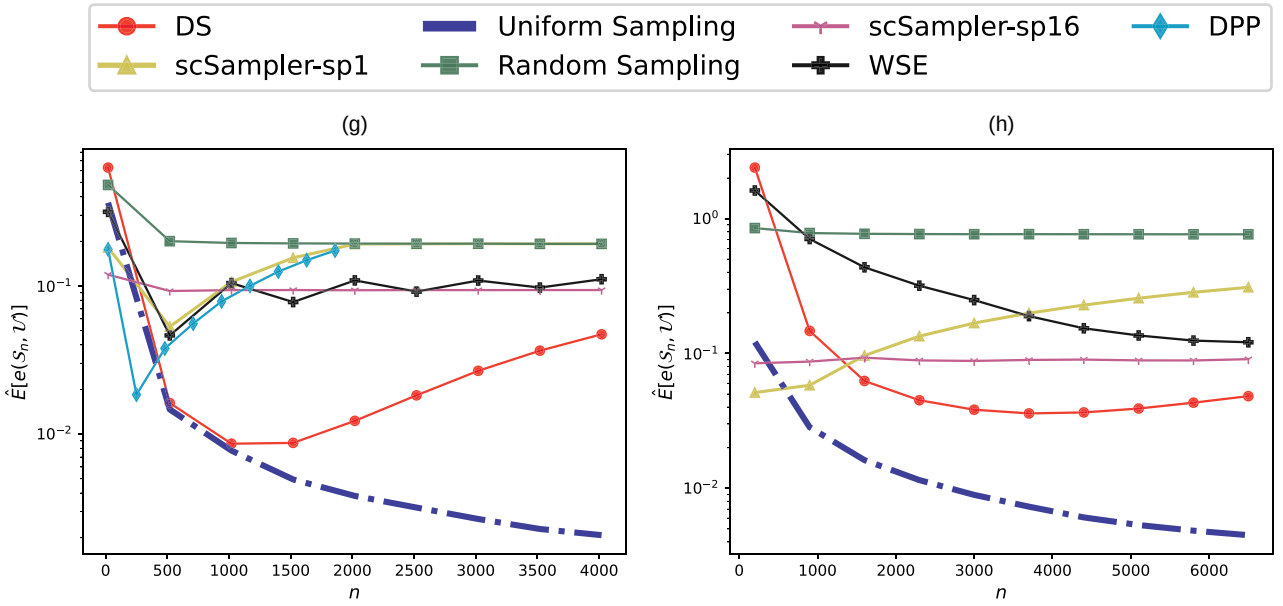


Figure 8. (Continued)



Notes. Uniform sampling serves as a reference (the closer the performance of an algorithm is to the Uniform Sampling curve, the better). (a) Normal, 2D. (b) Normal, 10D. (c) Gamma, 2D. (d) Gamma, 10D. (e) Exp, 2D. (f) Exp, 10D. (g) MGM, 2D. (h) MGM, 10D.

DPP and WSE were prohibitively slow for even the smallest size examples in Table 1 and were too slow to be included in the table for the larger size examples.

The DS algorithm mainly consists of three parts: estimating the density of D upfront from scratch (line 12 in Procedure 1), updating the density periodically (line 23 in Procedure 1) and selecting subsample points (line 15 and line 26 in Procedure 1). For convenience, we will refer these three parts to initial density estimation, density updating, and subsample selection, respectively. We observe that for all tested examples, initial density estimation and density Updating cost around 95% of the total computation time, and the computational costs of initial density estimation and density updating are roughly comparable (results are not shown here for brevity). Considering that the density estimation accounts for most of the computational cost, for large N the DS expense could be reduced by randomly sampling some fraction of D to use for the density estimation. This is somewhat similar to the computational strategy used by scSampler-sp16 to reduce computational expense relative to scSampler-sp1. The former randomly partitions D into 16 subsets and applies scSampler-sp1 to each subset and then combines the 16 subsamples.

6. Conclusions

This paper presents a novel diversity subsampling algorithm, the DS algorithm, that selects an i.i.d. uniform

Table 1. Runtime Comparisons of Subsampling Algorithms

	Subsample size	DS	scSampler-sp1	scSampler-sp16	DPP	WSE
$q = 10$, $N = 10^5$	$n = 1{,}000$	1.17×10^1	1.29×10^1	8.34×10^{-1}	NA	3.09×10^3
	$n = 8{,}000$	1.16×10^1	9.79×10^1	6.34×10^0	NA	3.05×10^3
	$n = 20{,}000$	1.14×10^1	2.33×10^2	1.52×10^1	NA	2.92×10^3
$q = 10$, $N = 10^6$	$n = 1{,}000$	1.17×10^2	1.59×10^2	7.84×10^0	NA	NA
	$n = 8{,}000$	1.17×10^2	1.29×10^3	6.15×10^1	NA	NA
	$n = 20{,}000$	1.15×10^2	3.01×10^3	1.60×10^2	NA	NA
$q = 20$, $N = 10^6$	$n = 1{,}000$	1.40×10^2	2.31×10^2	1.02×10^1	NA	NA
	$n = 8{,}000$	1.40×10^2	1.72×10^3	7.69×10^1	NA	NA
	$n = 20{,}000$	1.40×10^2	$3.42 \times 10^3+$	1.97×10^2	NA	NA
$q = 40$, $N = 10^6$	$n = 1{,}000$	1.73×10^2	3.36×10^2	1.32×10^1	NA	NA
	$n = 8{,}000$	1.73×10^2	$2.64 \times 10^3+$	1.02×10^2	NA	NA
	$n = 20{,}000$	1.74×10^2	NA	2.54×10^2	NA	NA

Notes. The runtime is reported in seconds and averaged over 100 replications. NA indicates that an subsampling algorithm took longer than one hour to finish for all tested replications. + on the right side of a number indicates the real statistic was no smaller than the reported statistic (replication runs that took longer than one hour were terminated and excluded from the average, in which case the reported numbers are somewhat lower than actual runtimes).

subsample from a data set over the effective support of the empirical data distribution to the largest extent possible. The asymptotic performance of the DS algorithm was presented for the case that $f(\boldsymbol{x})$ is known, and its effectiveness and advantages over existing diversity subsampling algorithms were demonstrated numerically for the intended situation in which $f(\boldsymbol{x})$ must be estimated: Overall, the DS algorithm selects subsamples more similar to a true i.i.d. uniform sample than existing algorithms do with much lower computational cost. We have also presented a generalized version of the DS algorithm to select subsamples following any desired target density (or nonnegative target function in general). All methods proposed in the paper are implemented in the FADS (Shang et al. 2022) Python package.

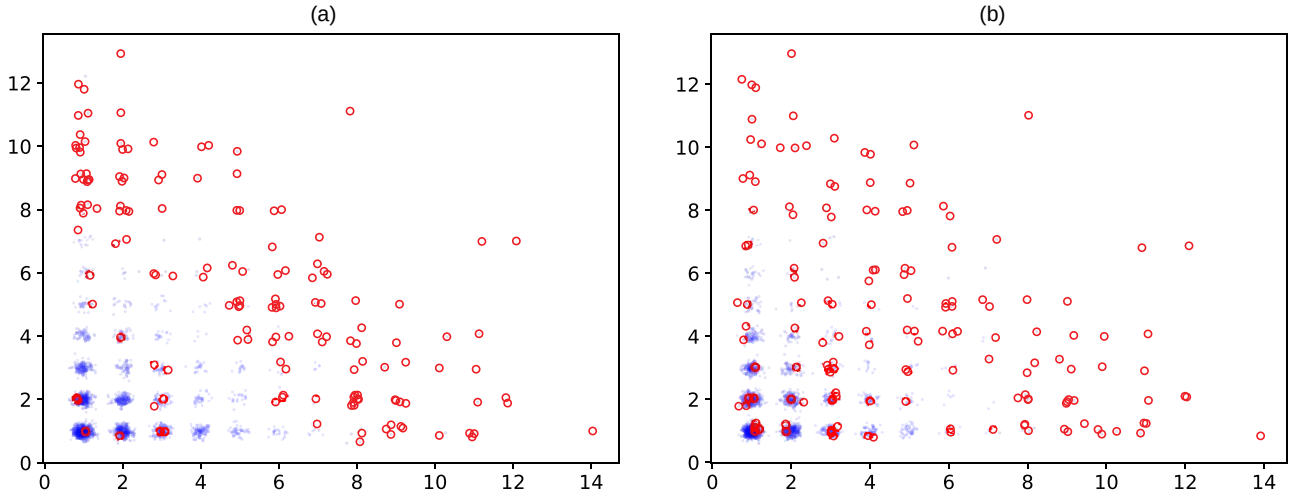
Appendix A. Numerical Comparisons of Using $\hat{f}(\boldsymbol{x}_i)$ vs. $\hat{f}(\tilde{\boldsymbol{x}}_i)$

Using the DS algorithm, a size $n = 150$ subsample was selected from the data shown in Figure A.1, for which $q = 2$ and $N = 10{,}000$. The data were generated from the bivariate geometric distribution with mean 2 and support $\{1, 2, \cdots\}$. In Figure A.1, a small dot and open circle represent a data point and a selected subsample point, respectively (jitter was added to all data for visualization). Comparing Figure A.1(b) to Figure A.1(a), we see that the DS subsample selected using the estimated density at the perturbed data, as opposed to the original data, appears to be much more diverse and uniformly distributed over the empirical data support. One possible explanation for this may be that the density estimated at the original data has larger bias. Consider the" mode" of the bivariate geometric data distribution at point $(1,1)$. Let $\{i_1, \ldots, i_k\}$ denote the indices for which $\boldsymbol{x}_{i_1} = \cdots = \boldsymbol{x}_{i_k} = (1,1)$, and the density estimates for these k data points form a set of density estimations at the mode $(1,1)$. After estimating $\hat{f}$ using the perturbed data, we can either use $\hat{f}$ evaluated on the original data points, that is, $T_1 = \{\hat{f}(\boldsymbol{x}_{i_1}), \ldots, \hat{f}(\boldsymbol{x}_{i_k})\}$ or on the perturbed data points $T_2 = \{\hat{f}(\tilde{\boldsymbol{x}}_{i_1}), \ldots, \hat{f}(\tilde{\boldsymbol{x}}_{i_k})\}$ to estimate the p.m.f value at the mode, that is, $f((1,1))$. To compare T_1 and T_2 as density estimates, we compute and compare their sample estimates of mean square errors, biases, and variances, respectively.[3] To be specific, for T_1, the sample estimates of mean square error, bias, and variance were computed using $1/k\sum_{t=1}^{k}\{\hat{f}(\boldsymbol{x}_{i_t}) - f((1,1))\}^2$, $1/k\sum_{t=1}^{k}\{\hat{f}(\boldsymbol{x}_{i_t}) - f((1,1)\}$ and $1/k\sum_{t=1}^{k}\{\hat{f}(\boldsymbol{x}_{i_t}) - 1/k\sum_{m=1}^{k}\hat{f}(\boldsymbol{x}_{i_m})\}^2$, respectively. For T_2, the sample estimates of mean square error, bias, and variance were computed using $1/k\sum_{t=1}^{k}\{\hat{f}(\tilde{\boldsymbol{x}}_{i_t}) - f((1,1))\}^2$, $1/k\sum_{t=1}^{k}\{\hat{f}(\tilde{\boldsymbol{x}}_{i_t}) - f((1,1)\}$ and $1/k\sum_{t=1}^{k}\{\hat{f}(\tilde{\boldsymbol{x}}_{i_t}) - 1/k\sum_{m=1}^{k}\hat{f}(\tilde{\boldsymbol{x}}_{i_m})\}^2$, respectively. Although T_1 had a much smaller mean square error (0.0013 for T_1 versus 0.0268 for T_2) and a variance of exactly zero, its estimated bias was 0.0364, whereas for T_2, the estimated bias was 0.0280. Similar phenomenon was observed when we repeated this experiment in 10D. This seems to suggest that estimating the density at the perturbed data instead of the original data reduced the bias of the p.m.f. estimation, which allowed the proposed DS algorithm to produce more diverse subsamples.

Appendix B. Density Estimation for the DS Algorithm

The DS algorithm (Procedure 1) can be used with any density estimation method, whether the estimated density integrates to 1, for instance, fixed or variable bandwidth KDE (Rosenblatt 1956, Parzen 1962, Terrell and Scott 1992), KNN density estimation (Mack and Rosenblatt 1979)), and Gaussian mixture model (GMM) density estimation (Reynolds 2009). Taking both density estimation accuracy and computational efficiency into consideration, we found that GMM density estimation with diagonal covariance matrices was overall the most effective for use in the DS algorithm. In all of our examples, we estimate the parameters of the

Figure A.1. (Color online) Subsamples Selected by the DS Algorithm for Using the Geometric Example at $q = 2$, By Using the Estimated Density at the Original Data Points (Left) vs. the Perturbed Data Points (Right)



Notes. The small dots represent the original data, and the open circles represent the selected subsample. Jitter was added to all data points for visualization. (a) Option 1: Estimating density at $\boldsymbol{x}_i$. (b) Option 2: Estimating density at $\tilde{\boldsymbol{x}}_i$.

GMM model using the popular expectation-maximization (EM) technique (Dempster et al. 1977, Reynolds 2009).

The GMM approach models the density as

$$f(\boldsymbol{x}) = \sum_{k=1}^{M} c_k f_k(\boldsymbol{x}|\boldsymbol{\theta}_k), \tag{B.1}$$

where $c_k \geq 0$, $\forall k = 1,\ldots,M$ and $\sum_{k=1}^{M} c_k = 1$. Here M is the number of components for the GMM model, and $f_k(\boldsymbol{x}|\boldsymbol{\theta}_k)$ is the Gaussian p.d.f. with mean $\boldsymbol{\mu}_k$ and a diagonal covariance matrix $\mathrm{Diag}(\sigma_{k1}^2,\ldots,\sigma_{kq}^2)$. We denote $\boldsymbol{\theta}_k = (c_k, \boldsymbol{\mu}_k, \sigma_{k1},\ldots,\sigma_{kq})$ and write

$$f_k(\boldsymbol{x}|\boldsymbol{\theta}_k) = \prod_{j=1}^{q} \frac{1}{\sqrt{2\pi}\sigma_{kj}} e^{-\frac{(x_j-\mu_{kj})^2}{2\sigma_{kj}^2}}. \tag{B.2}$$

Given initial guesses for $\boldsymbol{\theta}_1^{(0)},\ldots,\boldsymbol{\theta}_M^{(0)}$, we apply the EM algorithm a fixed number (denoted *niter*) of iterations to estimate the parameters $\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_M$. The algorithm is outlined as Procedure 2. See, for example Reynolds (2009), for detailed discussions and derivations for GMM density estimation.

Algorithm B.1 (GMM Density Estimation Using EM for the DS Algorithm (Procedure 1))

Input: N, q, $D = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$, *niter*, M, $\boldsymbol{\theta}_1^{(0)},\ldots,\boldsymbol{\theta}_M^{(0)}$

1: **for** $t \in \{1,\ldots,niter\}$ **do**
2: **for** $k \in \{1,\ldots,M\}$ **do**
3: **for** $i \in \{1,\ldots,N\}$ **do**
4: $p_{ik}^{(t-1)} \leftarrow \frac{c_k^{(t-1)} f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{(t-1)})}{\sum_{j=1}^{M} c_j^{(t-1)} f_j(\boldsymbol{x}_i|\boldsymbol{\theta}_j^{(t-1)})}$
5: **end for**
6: **for** $j \in \{1,\ldots,q\}$ **do**
7: $\mu_{kj}^{(t)} \leftarrow \frac{\sum_{i=1}^{k} p_{ik}^{(t-1)} x_{ij}}{\sum_{i=1}^{k} p_{ik}^{(t-1)}}$
8: $\sigma_{kj}^{(t)} \leftarrow \sqrt{\frac{\sum_{i=1}^{k} p_{ik}^{(t-1)} x_{ij}^2}{\sum_{i=1}^{k} p_{ik}^{(t-1)}} - \left(\mu_{kj}^{(t)}\right)^2}$
9: **end for**
10: $c_k^{(t)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} p_{ik}^{(t-1)}$
11: **end for**
12: **end for**
13: **for** $k = 1,\ldots,M$ **do**
14: **for** $i = 1,\ldots,N$ **do**
15: Compute $f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{(niter)})$ according to Equation (B.2)
16: **end for**
17: **end for**
18: **for** $i = 1,\ldots,N$ **do**
19: $\hat{f}_{\mathrm{GMM}}(\boldsymbol{x}_i) \leftarrow \sum_{k=1}^{M} c_k^{(niter)} f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{(niter)})$
20: **end for**

Output: Density estimation at data points in D, $\{\hat{f}_{\mathrm{GMM}}(\boldsymbol{x}_i)$, for $i = 1,2,\ldots,N\}$

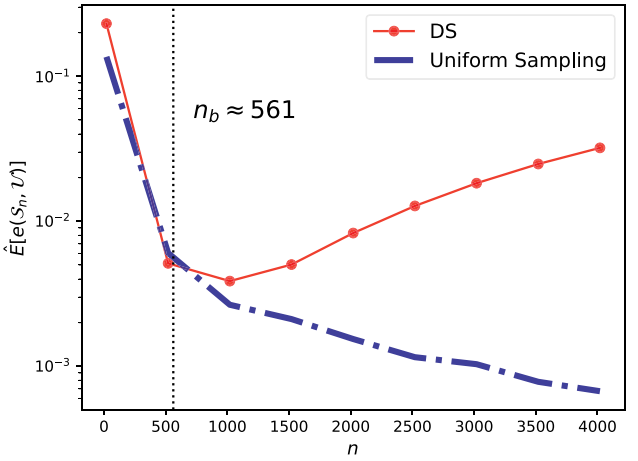
For all of our examples, we chose *niter* = 10 and $M = 32$ for the initial density estimation (line 12 of Procedure 1) of DS and used an existing implementation by Pedregosa et al. (2011) for the GMM density estimation. We set all GMM density estimation parameters as their default values except for *niter* and M. The density updating of the DS algorithm (line 23 of Procedure 1) is methodologically similar to the initial density estimation, except that for better efficiency, we use the estimated values of $\{\boldsymbol{\theta}_k\}_{k=1}^{M}$ from the previous update as the initial guesses in the current updating procedure, and we set *niter* = 1. For the density updating procedure, we fit the GMM model to only the data points in D that have not been previously selected.

Appendix C. Estimating the Deviation Point of the DS Algorithm

In this section, we take the DS subsample as an example to illustrate the point that for finite N, any subsampling algorithm will inevitably deviate from uniform sampling after a certain subsample size n, due to points in sparser regions being depleted from D. For a subsample of size n selected by the DS algorithm, denoted by $\mathcal{S}_n = \{\boldsymbol{x}_{j_1},\ldots,\boldsymbol{x}_{j_n}\}$, with $\{j_1,\ldots,j_n\} \subset \{1,\ldots,N\}$, we use $n_\Omega = \sum_{k=1}^{n} \mathbf{1}_\Omega(\boldsymbol{x}_{j_k})$ to denote the number of selected points in $\mathcal{S}_n$ lying inside Ω. Consider a small region $\mathrm{d}V$ in the lowest density region inside Ω and denote the lowest density function value of $f(\boldsymbol{x})$ inside Ω by $f_{\min,\,\Omega}$. Then among data points in D, there are, on average, approximately $Nf_{\min,\,\Omega}|\mathrm{d}V|$ data points inside region $\mathrm{d}V$. Similarly, if $\mathcal{S}_n$ is i.i.d. uniformly distributed over Ω, then $\mathcal{S}_n$ contains on average approximately $n_\Omega|\mathrm{d}V|/|\Omega|$ selected points inside region $\mathrm{d}V$. Therefore, the deviation from uniformity over Ω of a subsample theoretically must begin no later than when $n_\Omega|\mathrm{d}V|/|\Omega| = Nf_{\min,\,\Omega}|\mathrm{d}V|$, that is, when $n_\Omega = Nf_{\min,\ \Omega}|\Omega| \stackrel{\mathrm{def}}{=} n_\Omega^b$. We refer to the DS subsample size, n^b, at which there are n_Ω^b points inside Ω as the deviation point of the DS algorithm.

Figure C.1 shows an example with estimated $n^b \approx 561$ (denoted by the vertical dash line) using the normal example with $q = 2$. The experimental setup is the same as in Section 5.3. From Figure C.1, we see that the average $e(\mathcal{S}_n, \mathcal{U})$ for the DS algorithm starts to deviate from the results of uniform sampling at around n^b, as the previous theoretical arguments predict.

Figure C.1. (Color online) Estimation of n_b (Shown by the Vertical Dotted Line) for the DS Algorithm



Note. Uniform sampling serves as a reference (the closer the performance of an algorithm is to the Uniform Sampling curve, the better).

Figure D.1. (Color online) Averaged $r(\mathcal{S}_n)$ (Equation (11)), as a Function of n, for Subsamples Selected by DS, Random Sampling from D (Random Sampling), scSampler-sp1, scSampler-sp16, WSE, and DPP

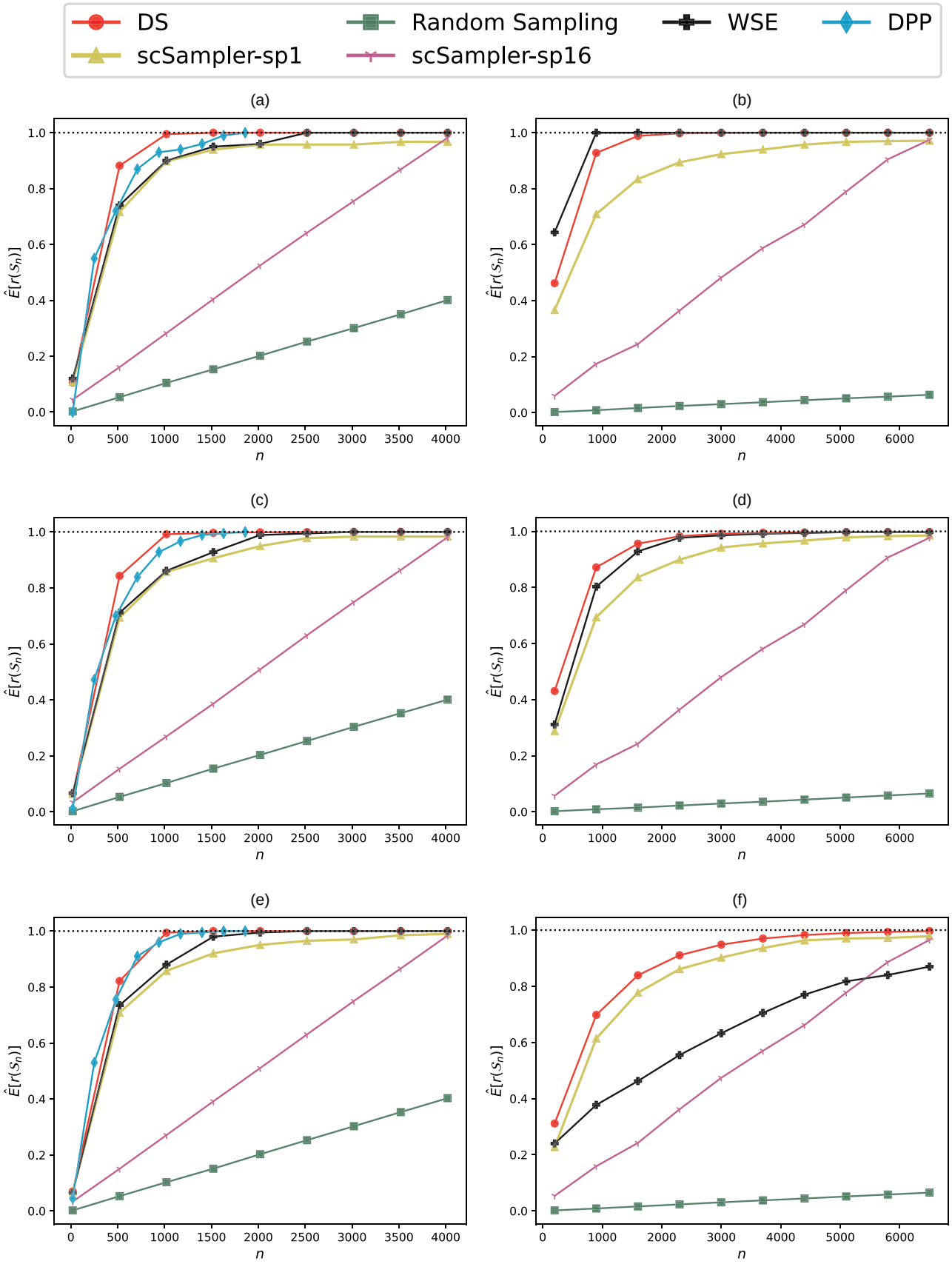
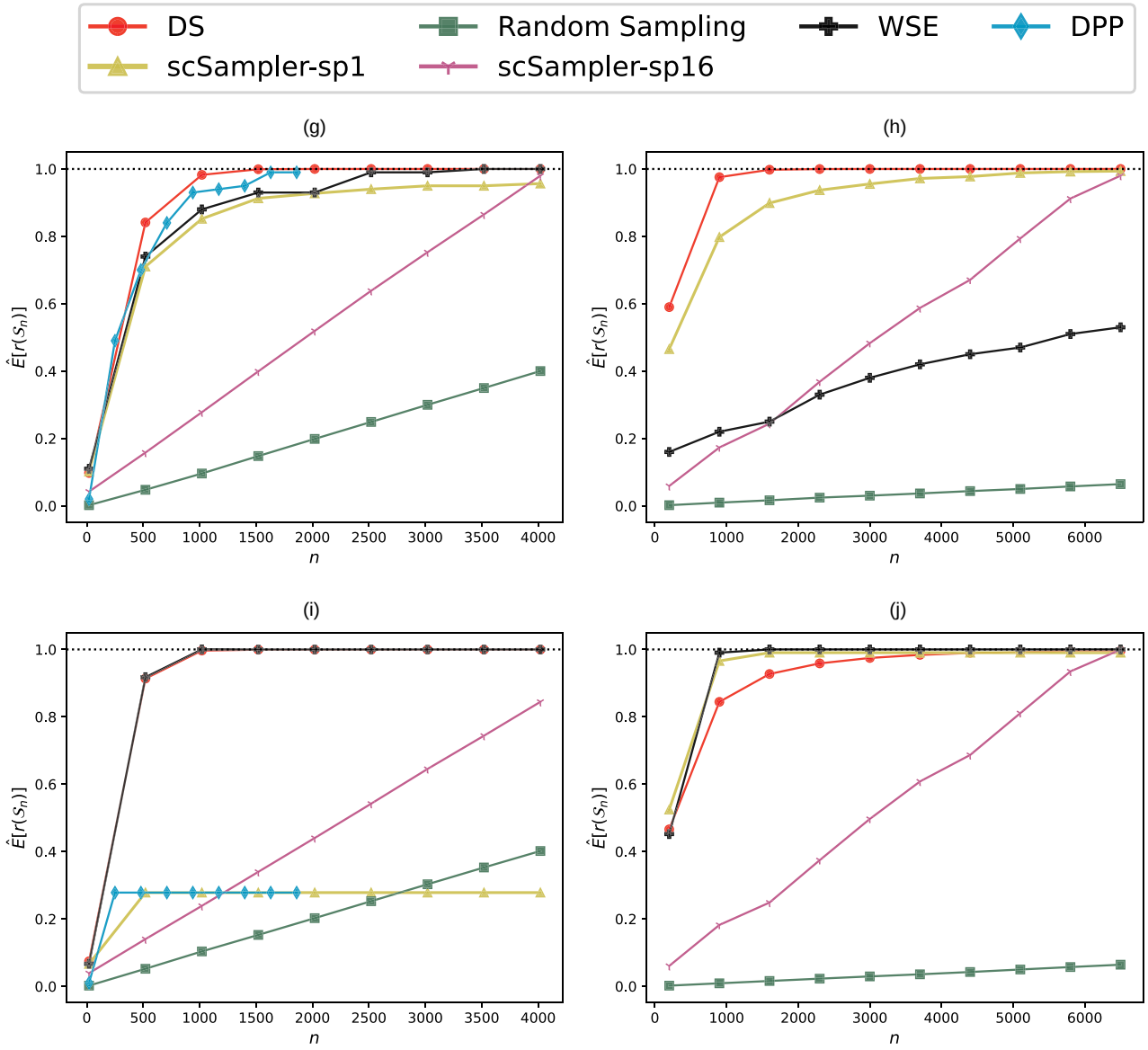


Figure D.1. (Continued)



Notes. Uniform Sampling serves as a reference (the closer the performance of an algorithm is to the Uniform Sampling curve, the better). (a) Normal, 2D. (b) Normal, 10D. (c) Gamma, 2D. (d) Gamma, 10D. (e) Exp, 2D. (f) Exp, 10D. (g) MGM, 2D. (h) MGM, 10D. (i) Geometric, 2D. (j) Geometric, 10D.

Appendix D. Numerical Performance Results for the Low-Density Region Sampling Ratio Criterion

Figure D.1 shows the average low-density region sampling ratio (Equation (11)) for the six subsampling algorithms, namely random sampling, DS, WSE, scSampler-sp1, scSampler-sp16, and DPP. The uniform sampling reference is irrelevant under this criterion, because it only generates samples within Ω. In 8 of 10 examples (Figure D.1, (a) and (c)–(i)), the average low-density region sampling ratio of the DS algorithm converges to one faster than (Figures D.1, (a), (c), (d), and (f)–(h)) or comparably to (Figure D.1, (e) and (i)) the other subsampling algorithms, which demonstrates its effectiveness for selecting data points in the low-density regions when n is small. For the other two examples (Figure D.1, (b) and (j)) performances of DS are not far behind the best performing method.

Analogous results for the replicated D examples are shown in Figure D.2, from which we see that the results of DS are not adversely affected much when the data sets have many replications, and DS remains among the best performing algorithms for all examples. For some examples (Figure D.2, (a), (c), (e), (g), and (h)), DS performs substantially better than the next best performing method.

Figure D.2. (Color online) Averaged $r(\mathcal{S}_n)$ (Equation (11)), as a Function of n, Using Replicated Data, for Subsamples Selected by DS, Random Sampling from D (Random Sampling), scSampler-sp1, scSampler-sp16, WSE, and DPP

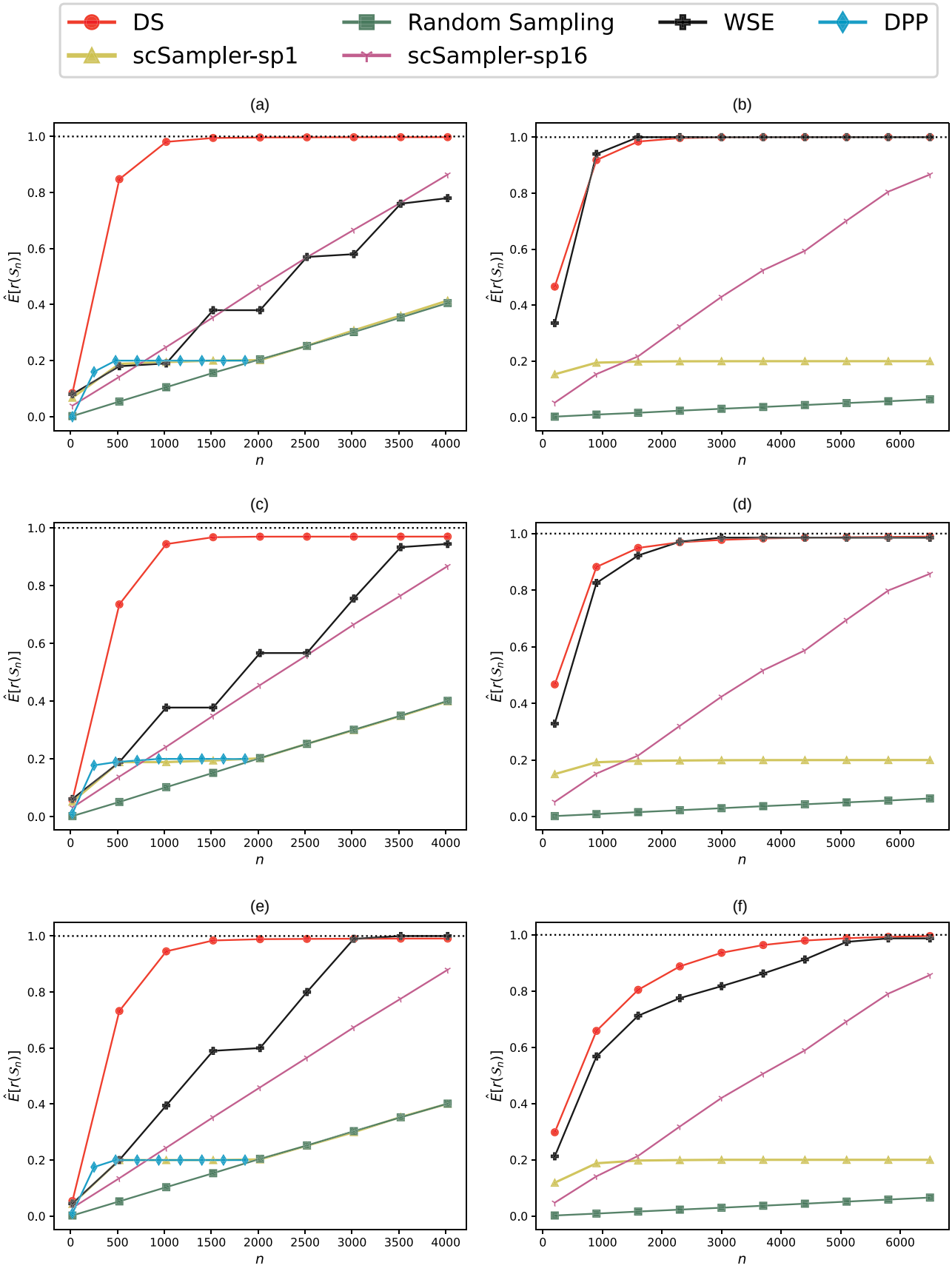
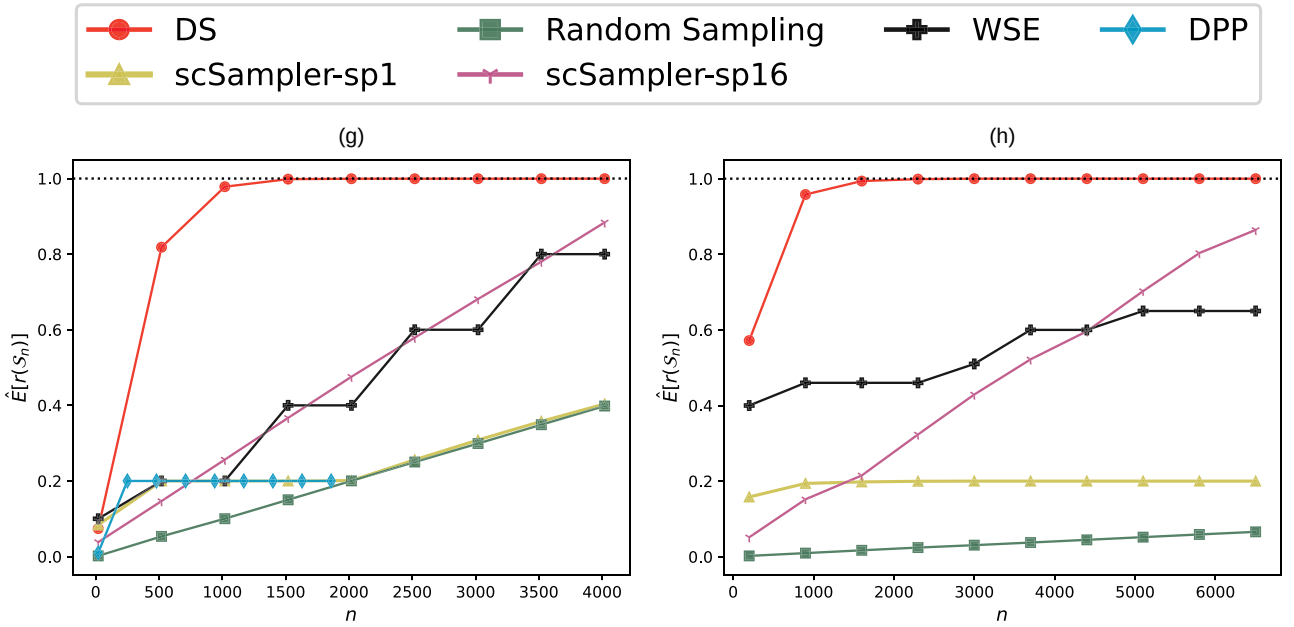


Figure D.2. (Continued)



Notes. Uniform Sampling serves as a reference (the closer the performance of an algorithm is to the Uniform Sampling curve, the better). (a) Normal, 2D. (b) Normal, 10D. (c) Gamma, 2D. (d) Gamma, 10D. (e) Exp, 2D. (f) Exp, 10D. (g) MGM, 2D. (h) MGM, 10D.

Endnotes

[1] For all subsampling algorithms, the subsample size inside Ω may be slightly less than n because n is the subsample size selected from the entire data set and some selected points might be outside of Ω.

[2] We wrapped the published WSE code (Yuksel 2016) in Python and tested its runtime directly from Python.

[3] The estimated density function was obtained using the Gaussian mixture model (Reynolds 2009); see Appendix B for details.

References

Biyik E (2019) dpp sampler.py. Accessed March 26, 2020, https://github.com/Stanford-ILIAD/DPP-Batch-Active-Learning/blob/master/classification_synthetic/dpp_sampler.py.

Biyik E, Wang K, Anari N, Sadigh D (2019) Batch active learning using determinantal point processes. Preprint, submitted June 19, https://arxiv.org/abs/1906.07975.

Chen Y, Zhang N (2022) Density regression with conditional support points. *Technometrics* 64(3):1–13.

Cook RL (1986) Stochastic sampling in computer graphics. *ACM Trans. Graphics* 5(1):51–72.

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statist. Soc. B* 39(1):1–22.

Fanaee-T H, Gama J (2014) Event labeling combining ensemble detectors and background knowledge. *Progress Artificial Intelligence* 2:113–127.

Gelman A, Carlin JB, Stern HS, Rubin DB (1995) *Bayesian Data Analysis* (Chapman and Hall/CRC Press, Boca Raton, FL).

Han I, Gillenwater J (2020) Map inference for customized determinantal point processes via maximum inner product search. Chiappa S, Calandra R, eds. *Proc. Internat. Conf. on Artificial Intelligence and Statist.* (PMLR, New York), 2797–2807.

Haussmann E, Fenzi M, Chitta K, Ivanecky J, Xu H, Roy D, Mittel A, et al. (2020) Scalable active learning for object detection. *Proc. IEEE Intelligent Vehicles Sympos.* (IEEE, New York), 1430–1435.

Huang C, Joseph VR, Mak S (2022) Population quasi-Monte Carlo. *J. Comput. Graphics Statist.* 31(3):1–14.

Joseph VR, Mak S (2021) Supervised compression of big data. *Statist. Analysis Data Mining* 14(3):217–229.

Kennard RW, Stone LA (1969) Computer aided design of experiments. *Technometrics* 11(1):137–148.

Ko CW, Lee J, Queyranne M (1995) An exact algorithm for maximum entropy sampling. *Oper. Res.* 43(4):684–691.

Mack Y, Rosenblatt M (1979) Multivariate k-nearest neighbor density estimates. *J. Multivariate Anal.* 9(1):1–15.

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. Le Cam LM, Neyman J, eds. *Proc. 5th Berkeley Sympos. on Math. Statist. and Probability*, vol. 1 (University of California Press, Downtown Oakland, CA), 281–297.

Mak S, Joseph VR (2018) Support points. *Ann. Statist.* 46(6A):2562–2592.

McCool M, Fiume E (1992) Hierarchical poisson disk sampling distributions. Fiume E, ed. *Proc. Conf. on Graphics Interface*, vol. 92 (Canadian Information Processing Society, Mississauga, ON, Canada), 94–105.

Parzen E (1962) On estimation of a probability density function and mode. *Ann. Math. Statist.* 33(3):1065–1076.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, et al. (2011) Scikit-learn: Machine learning in Python. *J. Machine Learn. Res.* 12:2825–2830.

Puzyn T, Mostrag-Szlichtyng A, Gajewicz A, Skrzyński M, Worth AP (2011) Investigating the influence of data splitting on the predictive ability of qsar/qspr models. *Structural Chemistry* 22(4):795–804.

Ren P, Xiao Y, Chang X, Huang PY, Li Z, Gupta BB, Chen X, et al. (2021) A survey of deep active learning. *ACM Comput. Survey* 54(9):1–40.

Reynolds DA (2009) Gaussian mixture models. *Encyclopedia Biometrics* 741:659–663.

Rizzo M, Szekely G (2022) Energy: E-statistics: Multivariate inference via the energy of data. https://CRAN.R-project.org/package=energy.

Rosenblatt M (1956) Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.* 27(3):832–837.

Rubin D (1987) 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. Amer. Statist. Assoc.* 82:544–546.

Rubin DB (1988) Using the sir algorithm to simulate posterior distributions. *Bayesian Statist.* 3:395–402.

Shang B, Apley DW, Mehrotra S (2022) Fast diversity subsampling from a data set. Accessed June 2, 2022, https://pypi.org/project/FADS/.

Silveira AL, Barbeira PJS (2022) A fast and low-cost approach for the discrimination of commercial aged cachaças using synchronous fluorescence spectroscopy and multivariate classification. *J. Sci. Food Agriculture* 102(11):4918–4926.

Skare Ø, Bølviken E, Holden L (2003) Improved sampling-importance resampling and reduced bias importance sampling. *Scandinavian J. Statist.* 30(4):719–737.

Song D, Xi NM, Li JJ, Wang L (2022a) scsampler. Accessed February 9, 2022, https://github.com/SONGDONGYUAN1994/scsampler.

Song D, Xi NM, Li JJ, Wang L (2022b) scSampler: Fast diversity-preserving subsampling of large-scale single-cell transcriptomic data. *Bioinformatics* 38(11):3126–3127.

Székely G (2003) E-statistics: The energy of statistical samples. Technical report, Bowling Green State University, Department of Mathematics and Statistics, Bowling Green, Ohio.

Székely GJ, Rizzo ML (2004) Testing for equal distributions in high dimension. *InterStat* 5:1–6.

Terrell GR, Scott DW (1992) Variable kernel density estimation. *Ann. Statist.* 20(3):1236–1265.

Wang Z, Garrett CR, Kaelbling LP, Lozano-Pérez T (2018) Active model learning and diverse action sampling for task and motion planning. Maciejewski AA, ed. *Proc. IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems* (IEEE, New York), 4107–4114.

Wu D (2018) Pool-based sequential active learning for regression. *IEEE Trans. Neural Network Learn. Systems* 30(5):1348–1359.

Yu H, Kim S (2010) Passive sampling for regression. *Proc. IEEE Internat. Conf. on Data Mining* (IEEE, New York), 1151–1156.

Yuksel C (2015) Sample elimination for generating poisson disk sample sets. *Comput. Graphics Forum* 34(2):25–32.

Yuksel C (2016) cysampleelim.h. Accessed September 17, 2020, https://github.com/cemyuksel/cyCodeBase/blob/master/cySampleElim.h.