Fast Stealthily Biased Sampling Using Sliced Wasserstein Distance

Yudai Yamamoto YAMAMOTO@AR.SANKEN.OSAKA-U.AC.JP
Osaka University, Osaka, Japan

Satoshi Hara SATOHARA@UEC.AC.JP
The University of Electro-Communications, Tokyo, Japan

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

Ensuring fairness is essential when implementing machine learning models in practical applications. However, recent research has revealed that benchmark datasets can be crafted as fake evidence of fairness from unfair models using a method called Stealthily Biased Sampling (SBS). SBS minimizes the Wasserstein distance to manipulate a fake benchmark so that the distribution of the benchmark closely resembles the true data distribution. This optimization requires superquadratic time relative to the dataset size, making SBS applicable only to small-sized datasets. In this study, we reveal for the first time that the risk of manipulated benchmark datasets exists even for large-sized datasets. This finding indicates the necessity of considering the potential for manipulated benchmarks regardless of their size. To demonstrate this risk, we developed FastSBS, a computationally efficient variant of SBS using the Sliced Wasserstein distance. FastSBS is optimized by a stochastic gradient-based method, which requires only nearly linear time for each update. In experiments with both synthetic and real-world datasets, we show that FastSBS is an order of magnitude faster than the original SBS for large datasets while maintaining the quality of the manipulated benchmark.[1]

Keywords: Wasserstein distance, Sliced Wasserstein distance, Fairness

1. Introduction

Machine learning models have been utilized in various decision-making scenarios, such as loan approvals (Sheikh et al., 2020) and employment decisions (Raghavan et al., 2020). However, these models sometimes produce different predictions based on sensitive attributes such as gender or race, leading to unfairness (Mehrabi et al., 2021). Such unfair machine learning models can cause disadvantages or discrimination against individuals who are the subjects of these predictions (Angwin et al., 2016). Ensuring the fairness of models is therefore an essential element when implementing machine learning models to real-world usage.

A typical approach for ensuring fairness is to use fairness auditing tools (Saleiro et al., 2018; Adebayo, 2016; Bellamy et al., 2018) for evaluating and disclosing the fairness of models. Examples of these tools include those that compute various fairness metrics for models (Saleiro et al., 2018), measure the importance of different input information to the model (Adebayo, 2016), and identify groups with sensitive attributes that are being treated unfairly (Bellamy et al., 2018).

1. Our code is available at https://github.com/yyd-Yudai/FastSBS

Fukuchi et al. (2020) considered another way of ensuring fairness by publishing a *benchmark dataset* which is a subset of dataset with models' decisions. By publishing the benchmark dataset, everyone can assess the fairness of the decisions using the aforementioned auditing tools. Unfortunately, Fukuchi et al. (2020) also reported that the benchmark dataset can be manipulated to mislead the auditing results. This means that even if the model and its decisions are unfair, one can construct a benchmark dataset that appears fair when evaluated with auditing tools. Moreover, such manipulation is almost impossible to detect when the benchmark dataset is constructed using the method called *Stealthily Biased Sampling* (SBS) (Fukuchi et al., 2020).

SBS manipulates the benchmark dataset by minimizing the Wasserstein distance between the original dataset and the benchmark dataset so that the two datasets to be almost indistinguishable. SBS is formulated as the minimum-cost flow problem, which can be solved in $\tilde{O}(N^{2.5})$ time when applied to a dataset of size N. This superquadratic time complexity indicates that SBS is not scalable to large datasets. Therefore, the risk of manipulated benchmark datasets currently exists only for small-sized datasets.

In this study, we reveal that for the first time that the risk of manipulated benchmark datasets exists even for large-sized datasets. This finding underscores the need to be aware of the potential for manipulated benchmark datasets regardless of their size. To demonstrate this risk, we developed FastSBS, a computationally efficient variant of SBS. FastSBS is optimized by using a stochastic gradient-based method which requires only $\tilde{O}(N)$ time in each update. In the experiments with both synthetic and real-world datasets, we show that FastSBS is an order of magnitude faster than the original SBS on large datasets.

2. Preliminaries

Data and Sampling Suppose that we want to assess the fairness of a model f. We denote the input features to the model f as $x \in \mathbb{R}^d$ and the sensitive attribute as $s \in \{0, 1\}$. The predicted label by the model for this input is $y = f(x) \in \{0, 1\}$, and a dataset consisting of N tuples of (x, s, y) is represented as $D = \{(x_i, s_i, y_i)\}_{i=1}^N$.

A benchmark dataset, which is a set of K data points sampled from the dataset D, is denoted as Z. When constructing Z according to some sampling distribution, the probability that the i-th data point (x_i, s_i, y_i) from D is included in Z is denoted as $\mathbb{P}((x_i, s_i, y_i) \in Z) = \mu_i$. When Z is constructed by uniform random sampling, this probability is $\mathbb{P}((x_i, s_i, y_i) \in Z) = \nu_i = \frac{K}{N}$.

Wasserstein Distance Wasserstein distance (Villani, 2009) is a distance between two probability distributions. In this study, we consider the Wasserstein distance for discrete probability distributions. Let $C_{i,j}$ and $\pi_{i,j}$ represent the transportation cost and the transportation amount from the i-th point with the probability mass μ_i to the j-th point with the mass ν_j, respectively. The transportation cost $C_{i,j}$ is non-negative $C_{i,j} \geq 0$, which is typically chosen as $C_{i,j} = \|x_i - x_j\|$ or its square. The transportation amount $\pi_{i,j}$ is non-negative $\pi_{i,j} \geq 0$, and it satisfies the mass preservation laws $\sum_{j=1}^N \pi_{i,j} = \mu_i$ and $\sum_{i=1}^N \pi_{i,j} = \nu_j$. We denote these constraints on $\pi_{i,j}$ by $\pi \in \Pi(\mu, \nu)$. The Wasserstein distance $\mathrm{W}(\mu, \nu)$ between

μ and ν is defined as the optimal value of the following optimization problem:

$$\mathrm{W}(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \sum_{i=1}^{N} \sum_{j=1}^{N} C_{i,j} \pi_{i,j}. \tag{1}$$

Wasserstein distance can be computed in polynomial time using the Sinkhorn algorithm (Cuturi, 2013) or by reducing it to the minimum-cost flow problem (Kovács, 2015).

Sliced Wasserstein Distance Wasserstein distance between the one-dimensional distributions can be computed in $O(N \log N)$ time (Rabin et al., 2011). Sliced Wasserstein distance (Rabin et al., 2011) leverages this property of the one-dimensional Wasserstein distance for efficient computation. Let $u \in \mathbb{R}^d$ be a uniformly random vector on the unit sphere of a d-dimensional space. The d-dimensional input feature $x \in \mathbb{R}^d$ is then projected onto one dimension using u as $z = u^\top x$. Wasserstein distance in the one-dimensional space projected by u is given by:

$$\mathrm{W}^u(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \sum_{i=1}^{N} \sum_{j=1}^{N} C^u_{i,j} \pi_{i,j}, \tag{2}$$

where the typical choice of the transportation cost C^u_{ij} is the Euclidean distance $|z_i - z_j|$. Sliced Wasserstein distance is then defined as the expectation of this projected Wasserstein distance over u:

$$\mathrm{SW}(\mu, \nu) = \mathbb{E}_u[\mathrm{W}^u(\mu, \nu)]. \tag{3}$$

3. Faking Fairness through Sampling

In this section, we review the existing methods for creating a fair benchmark dataset Z from an unfair dataset D by manipulating the sampling distribution. To fake fairness, the benchmark dataset Z needs to fulfill the following two requirements (Fukuchi et al., 2020).

(R1) Z is fair with respect to some fairness criteria.

(R2) The distribution of Z is close to the distribution of the original dataset D.

(R1) ensures that Z is successfully faking fairness. (R2) ensures that the faking remains undetected. Otherwise, if the distribution of Z is clearly different from that of D, a third party can detect that Z is unnatural and hence manipulated.

3.1. Case-Control Sampling

The simplest method to achieve (R1) is Case-Control Sampling (Mantel and Haenszel, 1959). Suppose that one wishes to create a fair Z in terms of Demographic Parity (Dwork et al., 2012). For each of the four combinations of sensitive attribute $s \in \{0, 1\}$ and predicted label $y \in \{0, 1\}$, let the number of data points included in Z be $k_Z(s, y)$. Then, the fairness with respect to DP is realized on Z if the following holds:

$$\frac{k_Z(0,1)}{k_Z(0,0) + k_Z(0,1)} = \frac{k_Z(1,1)}{k_Z(1,0) + k_Z(1,1)}. \tag{4}$$

Thus, any benchmark dataset Z satisfying (4) fulfills (R1). One can easily construct such Z by sampling instances from D using the following distribution:

$$\mu_i = \frac{k_Z(s_i, y_i)}{k_D(s_i, y_i)}, \tag{5}$$

where $k_D(s, y)$ is the number of data points in D with the sensitive attribute value s and the predicted label value y. One can naturally generalize this idea to other fairness metrics such as Equalized Odds (Awasthi et al., 2020) and Equal Opportunity (Arneson, 2018) by designing an appropriate k_Z. Note that Case-Control Sampling only cares about the first requirement (R1).

3.2. Stealthily Biased Sampling

Stealthily Biased Sampling (SBS) of Fukuchi et al. (2020) optimizes the sampling distribution μ so that it fulfills both (R1) and (R2). (R1) is fulfilled by appropriately designing k_Z as in (4), which is then expressed as the following constraint on μ[2]:

$$\sum_{(x_i, s_i, y_i) \in D : , s_i = s, , y_i = y} \mu_i = k_Z(s, y). \tag{6}$$

We denote μ satisfying this constraint as $\mu \in P(k)$.

SBS then realizes (R2) by minimizing the Wasserstein distance between the sampling with μ and the uniform random sampling with ν:

$$\min_{\mu} \mathrm{W}(\mu, \nu), \quad \text{s.t.} \quad \mu \in P(k). \tag{7}$$

Equation (7) is a linear programming problem which is linear with respect to both π and μ, and can be solved in $\tilde{O}(N^{2.5})$ time by reducing it to the minimum-cost flow problem (Fukuchi et al., 2020).

4. Proposed Method

In this section, we propose FastSBS, a computationally-efficient variant of SBS. FastSBS optimizes μ using a stochastic gradient-based method, which requires only $\tilde{O}(N)$ time for each update. Consequently, FastSBS can be applied to large datasets. This finding underscores the necessity of being aware of the potential for manipulated benchmark datasets, regardless of their size.

In FastSBS, we replace the Wasserstein distance in SBS (7) with a computationally more efficient Sliced Wasserstein distance (3) as follows:[3]

$$\text{(FastSBS)} \quad \min_{\mu} \mathrm{SW}(\mu, \nu), \quad \text{s.t.} \quad \mu \in P(k). \tag{8}$$

2. We show the case for DP as the illustrative example.
3. One can also consider another variant of SBS using Tree-Sliced Wasserstein distance (Le et al., 2019). See Appendix C.

Algorithm 1: FastSBS Using Stochastic Gradient

Procedure FastSBS ($D = \{(x_i, s_i, y_i)\}_{i=1}^N$, $\{\eta^{(\ell)} > 0\}_\ell$, $L \in \mathbb{N}$)

$\mu_i^{(0)} \leftarrow \frac{1}{N}, \forall i \in \{1, 2, \ldots, N\}$;

for $\ell = 1, 2, \ldots, L-1$ **do**

Randomly sample projection u;

Compute subgradient $g^{(\ell)}$ with respect to μ for (8);

$\mu^{(\ell+1)} \leftarrow$ Update $\mu^{(\ell)}$ by using the subgradient step $\eta^{(\ell)} g^{(\ell)}$ within $P(k)$;

end

return $\sum_{\ell=1}^L \eta^{(\ell)} \mu^{(\ell)} / \sum_{\ell=1}^L \eta^{(\ell)}$;

The stochastic gradient-based optimization method for (8) is shown in Algorithm 1. In the algorithm, we need two essential steps. First, we need an unbiased estimate of the subgradient g of the objective function with respect to μ. Second, we need to update μ using the subgradeint within the constraint $P(k)$ so that the updated μ satisfies $P(k)$. The time complexity of the algorithm therefore depends on the time complexity for the subgradient evaluation and the update operation. In the following subsections, we show that both subgradient and update can be computed in $\tilde{O}(N)$ time.

4.1. Update of μ within $P(k)$

When DP is adopted for the fairness metric of interest, the constraint $P(k)$ in (6) can be rewritten as

$$\sum_{(x_i, s_i, y_i) \in D:, s_i = s, , y_i = y} \hat{\mu}_i = 1, \tag{9}$$

where $\hat{\mu}_i = \mu_i / k_Z(s,y)$. That is, we can interpret $\{\hat{\mu}_i\}_{s_i=s, y_i=y}$ as an element of a probability simplex and $\hat{g}_i = k_Z(s,y) g_i$ as its subgradeint. A well-known method for updating a parameter in the probability simplex is the multiplicative weight update (Arora et al., 2012), which is given as

$$\hat{\mu}_i \leftarrow \frac{\hat{\mu}_i \exp\left(-\eta \hat{g}_i\right)}{\sum_{(x_i, s_i, y_i) \in D:\, s_i = s,\, y_i = y} \hat{\mu}_i \exp\left(-\eta \hat{g}_i\right)}, \tag{10}$$

where $\eta > 0$ is a learning rate. By multiplying both sides by $k_Z(s,y)$, we obtain the updating rule

$$\mu_i \leftarrow \frac{k_Z(s,y) \mu_i \exp\left(-\eta k_Z(s,y) g_i\right)}{\sum_{(x_i, s_i, y_i) \in D:\, s_i = s,\, y_i = y} \mu_i \exp\left(-\eta k_Z(s,y) g_i\right)}. \tag{11}$$

This update of μ can be computed in $O(N)$ time.

4.2. Subgradient of Sliced Wasserstein Distance

By the linearity of expectation, the subgradient of Sliced Wasserstein Distance $\mathrm{SW}(\mu, \nu)$ is the expectation of the subgradient of $\mathrm{W}^u(\mu, \nu)$. Thus, to obtain an unbiased subgradient of $\mathrm{SW}(\mu, \nu)$, we only need to compute the subgradient of $\mathrm{W}^u(\mu, \nu)$ for a randomly chosen u. Once u is given and fixed, $\mathrm{W}^u(\mu, \nu)$ is the Wasserstein distance in one-dimension, which can be efficiently computed in $\tilde{O}(N)$ time using sorting.

One possible approach[4] for computing the subgradient of the one-dimensional Wasserstein distance $\mathrm{W}^u(\mu,\nu)$ is to solve the dual problem of (2) (Peyré et al., 2019):

$$\max_{g^\mu,g^\nu\in\mathbb{R}^N} \sum_{i=1}^N \mu_i g_i^\mu + \sum_{j=1}^N \nu_j g_j^\nu, \quad \text{s.t.} \quad g_i^\mu + g_j^\nu \le C_{i,j}^u,\ \forall i,j\in\{1,2,\dots,N\}. \tag{12}$$

The solutions g^μ, g^ν of the problem (12) correspond to the subgradients of $\mathrm{W}^u(\mu,\nu)$ with respect to μ and ν, respectively (Boyd and Vandenberghe, 2004).

By utilizing the one-dimensional structure of (12), we can solve the problem in $\tilde{O}(N)$ time using Algorithm 2. In the algorithm, we first sort $z_i = u^\top x_i$ so that z_i are sorted in an increasing order. This sorting takes $O(N\log N)$ time and dominates the entire time complexity of the algorithm; the other parts can be executed in $O(N)$ time.

To derive the algorithm, we reparametrize g^μ, g^ν by using $h_i^\mu = g_i^\mu - g_{i-1}^\mu$ and $h_j^\nu = g_j^\nu - g_{j-1}^\nu$ for $i,j\ge 2$ so that

$$g_1^\mu = 0, \quad g_i^\mu = \sum_{k=2}^i h_k^\mu, \quad g_j^\nu = g_1^\nu + \sum_{\ell=2}^j h_\ell^\nu, \quad \forall i,j\in\{2,3,\dots,N\}.$$

We note that we can always choose $g_1^\mu = 0$ without loss of generality because the problem (12) remains the same for an operation $g_i^\mu \leftarrow g_i^\mu + a, g_j^\nu \leftarrow g_j^\nu - a$ for any constant a.[5] With this reparametrization, we can rewrite the dual problem (12) as

$$\max_{g_1^\nu\in\mathbb{R},h^\mu,h^\nu\in\mathbb{R}^{N-1}} g_1^\nu + \sum_{k=2}^N \bar{\mu}_k h_k^\mu + \sum_{\ell=2}^N \bar{\nu}_\ell h_\ell^\nu, \quad \text{s.t.} \quad g_1^\nu + \sum_{k=2}^i h_k^\mu + \sum_{\ell=2}^j h_\ell^\nu \le C_{i,j}^u, \tag{13}$$

where $\bar{\mu}_k = \sum_{i=k}^N \mu_i$ and $\bar{\nu}_\ell = \sum_{j=\ell}^N \nu_j$.

An important property of the problem (13) is that we can decompose it to a series of subproblems. First, we can solve the subproblem on g_1^ν:

$$\max_{g_1^\nu\in\mathbb{R}} g_1^\nu, \quad \text{s.t.} \quad g_1^\nu \le C_{1,1}^u.$$

In our problem setting, $C_{1,1}^u = |z_1 - z_1| = 0$. Hence, we immediately have $g_1^\nu = 0$. Second, suppose that h_k^μ and h_ℓ^ν are optimized for $k = 2,3,\dots,n-1$ and $\ell = 2,3,\dots,m-1$. Then, we can construct a subproblem on h_n^μ and h_m^ν as

$$\max_{h_n^\mu,h_m^\nu\in\mathbb{R}} u_{n,m} + \bar{\mu}_n h_n^\mu + \bar{\nu}_m h_m^\nu, \quad \text{s.t.} \quad v_{n,m} + h_n^\mu \le C_{n,m-1}^u,\ v_{n,m} + h_m^\nu \le C_{n-1,m}^u, \tag{14}$$

where $u_{n,m} = \sum_{k=2}^{n-1} \bar{\mu}_k h_k^\mu + \sum_{\ell=2}^{m-1} \bar{\nu}_\ell h_\ell^\nu$ and $v_{n,m} = \sum_{k=2}^{n-1} h_k^\mu + \sum_{\ell=2}^{m-1} h_\ell^\nu$. When $\bar{\mu}_n \ge \bar{\nu}_m$, it is desired to choose h_n^μ so that $\bar{\mu}_n h_n^\mu$ is maximized, which results in $h_n^\mu = C_{n,m-1}^u - v_{n,m}$. For

4. One can also use POT library with automatic differentiation to compute subgradient. In Appendix A, we show that our Algorithm 2 is faster than POT. (POT: https://pythonot.github.io/)
5. One can easily verify that $\sum_{i=1}^N \mu_i(g_i^\mu + a) + \sum_{j=1}^N \nu_j(g_j^\nu - a) = \sum_{i=1}^N \mu_i g_i^\mu + \sum_{i=1}^N \nu_j g_j^\nu + a\underbrace{\left(\sum_{i=1}^N \mu_i - \sum_{j=1}^N \nu_j\right)}_{=0}$ and $(g_i^\mu + a) + (g_j^\nu - a) = g_i^\mu + g_j^\nu$.

Algorithm 2: Solving the dual of the one-dimensional Wasserstein distance (12)

Procedure Subgrad_1D $(D = \{(x_i, s_i, y_i)\}_{i=1}^N,\ \mu, \nu \in \mathbb{R}_+^N,\ u \in \mathbb{R}^d)$

$\sigma = \mathrm{argsort}\{z_i = u^\top x_i\}_{i=1}^N$ ▷ Sort after one-dimensional projection.

$\bar{\mu}_i \leftarrow \sum_{k=i}^N \mu_{\sigma(i)}$; $\bar{\nu}_i \leftarrow \sum_{k=i}^N \nu_{\sigma(i)}$; $h_i^\mu \leftarrow 0$; $h_i^\nu \leftarrow 0$; $\forall i \in \{2, 3, \ldots, N\}$

$n \leftarrow 2$; $m \leftarrow 2$; $v \leftarrow 0$;

while $n \le N$ and $m \le N$ **do**

  if $\bar{\mu}_n \ge \bar{\nu}_m$ **then** $h_n^\mu \leftarrow C_{n,m-1}^u - v$; $v \leftarrow C_{n,m-1}^u$; $n \leftarrow n+1$;

  else $h_m^\nu \leftarrow C_{n-1,m}^u - v$; $v \leftarrow C_{n-1,m}^u$; $m \leftarrow m+1$;

end

while $n \le N$ **do** $h_n^\mu \leftarrow C_{n,m-1}^u - v$; $v \leftarrow C_{n,m-1}^u$; $n \leftarrow n+1$;

while $m \le N$ **do** $h_m^\nu \leftarrow C_{n-1,m}^u - v$; $v \leftarrow C_{n-1,m}^u$; $m \leftarrow m+1$;

$g_{\sigma(1)}^\mu \leftarrow 0$; $g_{\sigma(i)}^\mu \leftarrow \sum_{k=2}^i h_k^\mu$; $\forall i \in \{2, 3, \ldots, N\}$ ▷ Subgradients in the original order

$g_{\sigma(1)}^\nu \leftarrow 0$; $g_{\sigma(j)}^\nu \leftarrow \sum_{\ell=2}^j h_j^\nu$; $\forall i \in \{2, 3, \ldots, N\}$

return g^μ, g^ν ▷ g^ν can be skipped when only g^μ is necessary.

this choice of h_n^μ, we also obtain the updated $v_{n+1,m}$ as $v_{n+1,m} \leftarrow v_{n,m} + (C_{n,m-1}^u - v_{n,m}) = C_{n,m-1}^u$. Similarly, when $\bar{\mu}_n < \bar{\nu}_m$, we have $h_m^\nu = C_{n-1,m}^u - v_{n,m}$ and $v_{n,m+1} \leftarrow C_{n,m+1}^u$. We can then update $n \leftarrow n+1$ or $m \leftarrow m+1$ to construct a new subprolem (14) on the updated n and m. By solving the subproblem (14) until $n = N$ and $m = N$ are met, we obtain the optimal h^μ and h^ν, and, hence, the optimal g^μ and g^ν.

4.3. Accelerating FastSBS using Fixed Slicing

The time complexity of FastSBS is dominated by the sorting operation. The other parts of the subgradient computations can be executed in $O(N)$ time. In theory, we need independent subgradients in every iteration for valid stochastic optimization. Thus, in every iteration of the optimization, we need to sample an independent projection u, incurring the cost of sorting for the new u.

Here, we consider a practical heuristic that can avoid the computational bottlenecks above. The idea is to *not* to sample the projections in every iteration. Instead, we sample sufficiently many projections before running FastSBS. That is, we first sample S projections $u_1, u_2, \ldots, u_S$ independently. For these projections, we compute the sorting of $z = u^\top x$ and store the results. In every optimization step, we randomly choose one projection among S candidates, and load the sorting result for the chosen one. This approach allows us to compute the subgradient in $O(N)$ time in every iteration, thereby resolving the computational bottleneck. Note that this heuristic corresponds to replacing the expectations of the Sliced Wasserstein distance (3) with a finite sample average.

$$\mathrm{SW}(\mu, \nu) \approx \frac{1}{S} \sum_{s=1}^S \mathrm{W}^{u_s}(\mu, \nu).$$

Our experimental results in Section 6 show that this heuristics is effective in practice. In the experiments, we observed that $S = 10$ is sufficient to attain a comparative performance with the independent random sampling of projections in every iteration. With the heuristics, we can enjoy the significant speedup of FastSBS without sacrificing its performance.

5. Related Work

The possibility of fake-fairness is studied in the intersection of fairness and explainability. Slack et al. (2020) and Merrick and Taly (2020) have shown that local explanation methods, such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), can be manipulated to produce fairer explanations even from unfair models. In the context of counterfactual explanation, Laugel et al. (2019) have shown that one can generate fairer counterfactuals. Aïvodji et al. (2019, 2021) have shown that global explanation methods can be manipulated as well.

In line with these studies, the study of Stealthily-Biased Sampling (SBS) (Fukuchi et al., 2020) can be categorized as the manipulation of the example-based explanations. Laberge et al. (2023) applied SBS to SHAP, enabling not only the manipulation of SHAP explanations but also the creation of fake benchmark datasets, making the results of the manipulated SHAP reproducible from the benchmark.

The risk of these manipulations depends on whether they can be implemented at a reasonable cost. If the cost of manipulation is prohibitively high, such methods are less likely to be used, indicating a lower risk. Prior studies on SBS (Fukuchi et al., 2020; Laberge et al., 2023) have demonstrated the capability of manipulation on relatively small datasets due to their superquadratic time complexity. These results suggest that SBS is not scalable to large datasets, and hence, the risk of manipulated benchmark datasets is mostly limited to small-sized datasets.

In the current study, however, we show that the risk of manipulated benchmark datasets exists even for large-sized datasets by designing FastSBS, a computationally efficient variant of SBS. This finding highlights the need to consider the potential for manipulated benchmark datasets regardless of their size.

6. Experiments

We compared the proposed FastSBS with existing methods using both synthetic and two real-world datasets, COMPAS (Angwin et al., 2016) and Adult (Dheeru and Taniskidou, 2017), following Fukuchi et al. (2020). In Sections 6.1 and 6.2, we show that FastSBS can generate benchmark with comparative performance as SBS. In Section 6.3, we demonstrate that FastSBS can be executed in far smaller runtime than SBS, while maintaining its performance. All the experiments were conducted on a workstation with Windows 10 Home 64bit OS, Intel Core i7-9700K CPU, and 32GB of RAMs.

Sampling Methods We adopted Case-Control Sampling (CC) (Mantel and Haenszel, 1959) and Stealthily Biased Sampling (SBS) (Fukuchi et al., 2020) as the baseline methods to be compared with. For these methods as well as for FastSBS, we set the number of instances in the benchmark Z for each pair of $(s, y) \in \{0, 1\} \times \{0, 1\}$ to satisfy

$$k_Z(0,0) = (1-\alpha)^2 K, \quad k_Z(0,1) = \alpha(1-\alpha)K, \quad k_Z(1,0) = \alpha(1-\alpha)K, \quad k_Z(1,1) = \alpha^2 K, \quad (15)$$

for several different values of $\alpha \in [0,1]$ so that the demographic parity (Dwork et al., 2012) of the sampled Z to be small.

For FastSBS, we set the number of fixed slices $S = 10$ in the heuristics. We set the learning rate at step ℓ to $\eta^{(\ell)} = 1/K\sqrt{\ell}$ for the synthetic and COMPAS datasets, and to $\eta^{(\ell)} = 1/3K\sqrt{\ell}$ for the Adult dataset. We adopted the following stopping criterion so that the optimization to terminate when the output of Algorithm 1 is sufficiently stable:

$$\sum_{i=1}^{N} |\bar{\mu}_i{}^{(t)} - \bar{\mu}_i{}^{(t+1)}| < \varepsilon, \tag{16}$$

where $\bar{\mu}^{(t)} = \sum_{\ell=1}^{t} \eta^{(\ell)}\mu^{(\ell)} / \sum_{\ell=1}^{t} \eta^{(\ell)}$. We set $\varepsilon = 0.01$ for the synthetic and COMPAS dataset experiments while we set $\varepsilon = 0.1$ for the Adult dataset experiment.

Evaluation Metrics In the experiments, we verified whether the benchmark dataset Z created by each method satisfies the requirements (R1) and (R2). As the fairness metric for the requirement (R1), we used Demographic Parity (Dwork et al., 2012) defined by

$$\mathrm{DP}(Z) = \left| \frac{k_Z(0,1)}{k_Z(0,0)+k_Z(0,1)} - \frac{k_Z(1,1)}{k_Z(1,0)+k_Z(1,1)} \right|. \tag{17}$$

Hence, the benchmark dataset Z with smaller $\mathrm{DP}(Z)$ is considered ideal. For assessing the requirement (R2), we used the Kolmogorov-Smirnov test (Massey Jr, 1951) for the synthetic data experiment in Section 6.1 and Wasserstein distance for the real-world data experiments in Section 6.2.

6.1. Experiment 1: Synthetic Data

Dataset Description We considered the following synthetic model (Fukuchi et al., 2020) that determines $y \in \{0,1\}$ from a uniformly random $x \in [0,1]$ and a uniformly random $s \in \{0,1\}$:

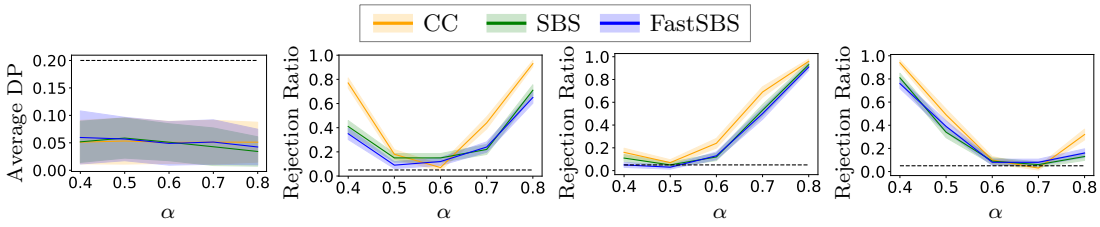
$$y = \mathbb{I}(x + 0.2s > 0.5). \tag{18}$$

In this model, the probability of $y = 1$ is 0.2 higher when $s = 1$ compared to that of $s = 0$. Therefore, the demographic parity of this model is 0.2. We generated a dataset D with $N = 1,000$ independent instances of (x, s, y) from this setup. For the purpose of assessing the Requirement (R2), we also generated another dataset D' with $N' = 200$ instances independently from D.

Evaluation of (R2) In the experiment, we sampled the benchmark dataset Z of size $K = 200$ from the dataset D using each of the sampling methods. We used the Kolmogorov-Smirnov test to compare the distribution of Z and D' for assessing the requirement (R2). Specifically, we conducted three statistical tests with null hypothesises $(H1)$–$(H3)$:

$$(H_1)\ p_{D'}(x) = p_Z(x), \quad (H_2)\ p_{D'}(x|y=0) = p_Z(x|y=0), \quad (H_3)\ p_{D'}(x|y=1) = p_Z(x|y=1),$$

where $p_{D'}$ and p_Z denote the distributions of x in the datasets D' and Z, respectively. If the test is frequently rejected for the sampled benchmark Z, the corresponding sampling method is considered to fail the requirement (R2) because the distributions of Z is found to be different from the independent dataset D'. We set the significance level of the tests to 0.05.



(a) Demographic Parity (b) Test on (H_1) (c) Test on (H_2) (d) Test on (H_3)

Figure 1: [Experiment 1] Comparison of the sampling methods on the synthetic dataset. The solid lines represent the average over 100 runs, and the shaded areas represent the standard deviation. The black dashed line in (a) indicates the original DP of 0.2, and the black dashed lines in (b)–(d) indicate the significance level 0.05.

Results Figure 1 shows the results of the 100 runs of the experiments. Figure 1(a) show that DP of each method is far lower than the original DP of 0.2. This result confirms that all the methods attained the requirement (R1) for any choice of α. Figures 1(b)–(d) show the results of the three statistical tests. In each figure, the vertical axis denotes the rejection ratio which is a fraction of the rejection over the 100 runs. Because we set the significance level to 0.05, the results with the rejection ratio around 0.05 indicate that the test could not distinguish D' and Z with sufficient significance. The figures show that SBS and the proposed FastSBS achieve rejection ratios close to 0.05 for around $\alpha = 0.6$, confirming that these methods fulfill the requirement (R2).

6.2. Experiment 2: COMPAS and Adult Data

COMPAS Dataset For the first real-world dataset, we used the COMPAS dataset (Angwin et al., 2016). Following the preprocessing steps used in the ProPublica analysis (Angwin et al., 2016), we obtained $x \in \mathbb{R}^8$, $s \in \{0, 1\}$, and $y \in \{0, 1\}$. In the COMPAS dataset, the probability of $y = 0$ for $s = 1$ is approximately 0.24 higher than that of $s = 0$, resulting in DP of 0.24. From the entire dataset of 5,278 instances, we randomly sampled 4,000 instances to create the dataset D and left the remaining 1,278 instances as the dataset D' for assessing (R2). We sampled $K = 1,000$ instances from D to construct the benchmark dataset Z using each sampling method.

Adult Dataset For the second real-world dataset, we used the Adult dataset (Dheeru and Taniskidou, 2017). In the preprocessing, categorical variables were converted to numerical variables[6], resulting in $x \in \mathbb{R}^{13}$, $s \in \{0, 1\}$, and $y \in \{0, 1\}$. From the entire dataset of 48,842 instances, we randomly sampled 10,000 instances as the training data and 20,000 instances as the test data, with the remaining 18,842 instances forming the dataset D' for assessing (R2). We trained linear logistic regression and random forest with 100 trees using the training data. We then labeled each of the test instances using the trained models and

6. We used the implementation from https://www.kaggle.com/code/kost13/us-income-logistic-regression/notebook.

constructed the dataset D of size $N = 20,000$. We sampled $K = 2,000$ instances from D to construct the benchmark dataset Z using each sampling method.

In the Adult dataset experiment, we used the bootstrap version of SBS (Fukuchi et al., 2020) as the baseline because the original SBS tends to be computationally prohibitive for the datasets D larger than $10,000$. In the bootstrap version, we first randomly sample 4,000 instances from D and then estimate μ using SBS on this 4,000 instances. We repeat this procedure 30 times, and use the average of the estimated μ for sampling. We parallelized this procedure to speed up SBS.

Evaluation of (R2) To assess the requirement (R2), we used the Wasserstein distance between two datasets $A = \{(x_i, s_i, y_i)\}_{i=1}^{N_A}$ and $B = \{(x'_j, s'_j, y'_j)\}_{j=1}^{N_B}$ defined as follows:

$$\mathrm{W}(A, B) = \min_{\pi \in \mathbb{R}^{N_A \times N_B}} \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \|x_i - x'_j\|^2 \pi_{i,j}, \quad \text{s.t.} \quad \sum_{i=1}^{N_A} \pi_{i,j} = \frac{1}{N_B}, \ \sum_{j=1}^{N_B} \pi_{i,j} = \frac{1}{N_A}.$$

Specifically, we evaluated three Wasserstein distances $\mathrm{W}(D', Z)$, $\mathrm{W}(D'_{y=0}, Z_{y=0})$, and $\mathrm{W}(D'_{y=1}, Z_{y=1})$. Here, $D'_{y=0} = \{(x, s, y) \in D \mid y = 0\}$ and $Z_{y=0} = \{(x, s, y) \in Z \mid y = 0\}$ denotes the subsets of D' and Z with $y = 0$, respectively. $D'_{y=1}$ and $Z_{y=1}$ are defined analogously. If all the three Wasserstein distances are small, the corresponding sampling method is considered to fulfill the requirement (R2) because the distributions of Z is found to be similar to that of the independent dataset D'.

Results Figures 2 shows the results of 100 runs of the experiments. In Figures 2(a), (e), and (i), we can observe that all the sampling methods could induce benchmark Z with small DP indicating all the methods fulfilled the requirements (R1). Figures 2(b)–(d), (f)–(h), and (j)–(l) confirm that the Wasserstein distance for SBS and the proposed FastSBS remain small for an appropriately chosen α, satisfying the requirement (R2).

6.3. Runtime Comparison

In our experiments, two hyper-parameters are governing the running time of the proposed FastSBS, which are the stopping threshold ε and the size of fixed slicing S. Larger ε leads to earlier stopping of optimization, while smaller S can reduce the computational overheads of sorting. Here, we show that, by appropriately choosing ε and S, the proposed FastSBS can be significantly faster than the original SBS while maintaining the quality of the sampled benchmark Z.

6.3.1. The Effects of ε

Setups We compared the runtime of FastSBS with the original SBS on the synthetic and Adult datasets.[7] For the synthetic and Adult datasets, we fixed the size of the benchmark Z to be $K = 200$ and $K = 2,000$, respectively, and varied the size of the dataset D.

7. Because the size of COMPAS is not large, we did not use it for the runtime comparison.

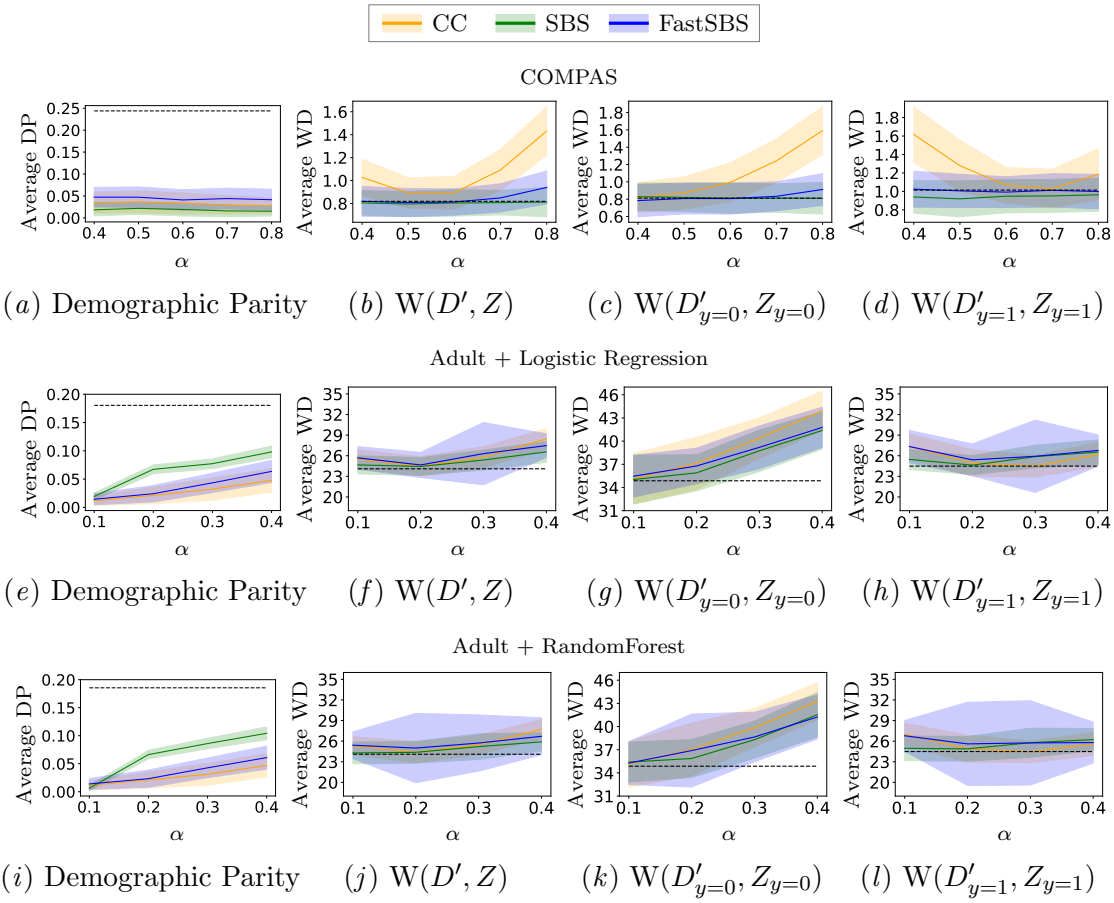


Figure 2: [Experiment 2] The results on COMPAS (a)–(d), Adult + Logistic Regression (e)–(h), and Adult + RandomForest (i)–(l). The solid lines represent the average over 100 runs, and the shaded areas represent the standard deviation. The black dashed lines in (a), (e), and (i) indicates the original DP, and the black dotted lines in (b)–(d), (f)–(h), and (j)–(l) represent the Wasserstein distance between the uniformly random sampled dataset from D and D'.

Result Figure 3 shows the average runtime on the synthetic dataset for each method over 10 runs for several different sizes of D. In the figure, we varied the threshold ε in the stopping criterion (16) to several different values. The figure indicates that the runtime of the proposed method is significantly faster than that of SBS unless we set ε too small. From these results, we can confirm that FastSBS can resolve the scalability issue of the original SBS. With FastSBS, one can construct a fake benchmark Z even from large D.

One may wonder whether using the larger ε for the speedup can result in early stopping of the optimization and the quality of the estimated μ can be poor. Figure 4 shows that this is not the case in our synthetic data experiment.[8] In the figures, we can find that FastSBS could sample the benchmark Z with sufficiently small DP as well as rejection

8. In Appendix B, we show the results on the COMPAS and Adult datasets.

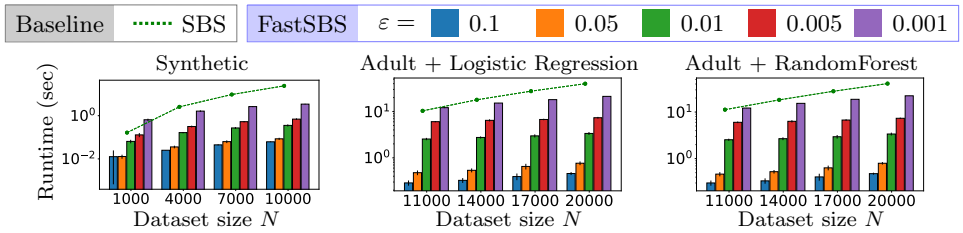
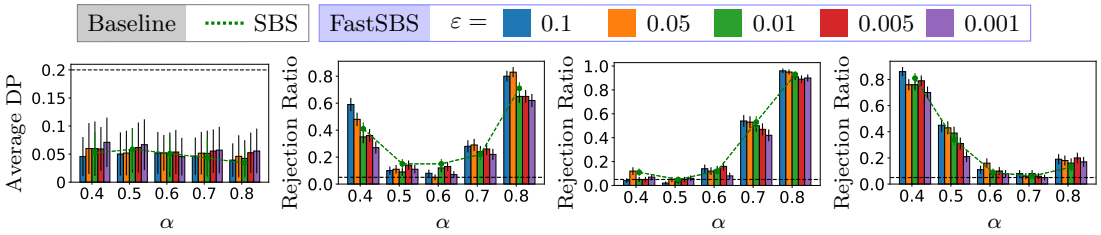


Figure 3: Runtimes of FastSBS for different choices of ε on the synthetic and Adult datasets.



(*a*) Demographic Parity (*b*) Test on (H_1) (*c*) Test on (H_2) (*d*) Test on (H_3)

Figure 4: The effects of ε on FastSBS on the synthetic data experiments.

ratios irrespective to the choice of ε. That is, the choice of ε is not that crucial in practice: we can enjoy the speedup without sacrificing the quality of the sampled Z.

6.3.2. The Effects of S

We compared the runtime of the proposed FastSBS with the original SBS by varying the the number of fixed slices S in our heuristics.

Result Figure 5 shows the average runtime on the synthetic dataset for each method over 10 runs for several different sizes of D. In the figure, we varied the sampling size S in the proposed heuristics to several different values. The figure indicates that the runtime of FastSBS is significantly faster than that of SBS for any S.

Figure 6 assesses the quality of the sampled benchmark Z for several different choice of S.[9] In the figures, we can find that the choice of S does not have significant impacts on the quality of Z. The choice of S is therefore not that crucial in practice and we can enjoy the speedup without sacrificing the quality of the sampled Z.

7. Conclusion

In this study, we demonstrate that it is possible to manipulate fake benchmarks even from large datasets. Previously, such manipulation was feasible only for small-sized datasets due to the superquadratic complexity of Stealthily Biased Sampling (SBS). However, our FastSBS (FastSBS) runs in $\tilde{O}(N)$ time per iteration, making it applicable to large datasets. This finding highlights the necessity of being aware of the potential for manipulated benchmark datasets regardless of their size. Our experimental results with both synthetic and

9. In Appendix B, we show the results on the COMPAS and Adult datasets.

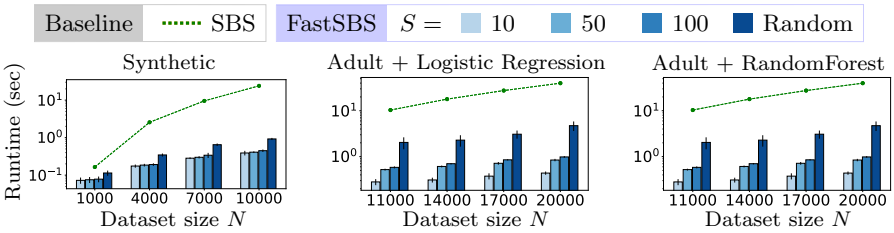
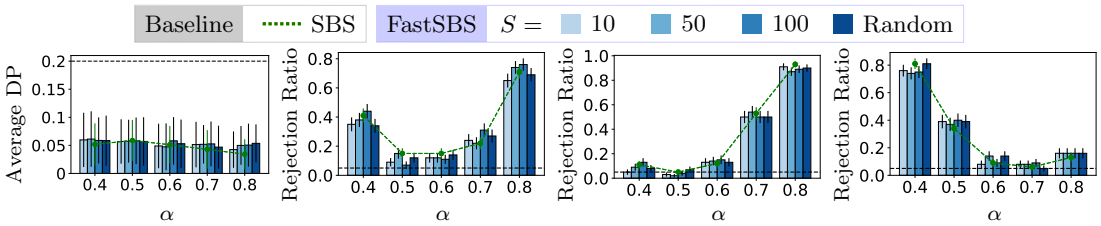


Figure 5: Runtimes of FastSBS for different choices of S on the synthetic and Adult datasets. Random does not use the proposed heuristics: it uses a new random u in every iteration of the optimization.



(*a*) Demographic Parity (*b*) Test on (H_1) (*c*) Test on (H_2) (*d*) Test on (H_3)

Figure 6: The effects of S on FastSBS on the synthetic data experiments. Random does not use the proposed heuristics: it uses a new random u in every iteration of the optimization.

real-world datasets show that FastSBS is an order of magnitude faster than the original SBS on large datasets while maintaining the quality of the sampled benchmark.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 23K28146.

References

Julius A Adebayo. *FairML: ToolBox for diagnosing bias in predictive modeling*. PhD thesis, Massachusetts Institute of Technology, 2016.

Ulrich Aïvodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*, pages 161–170, 2019.

Ulrich Aïvodji, Hiromi Arai, Sébastien Gambs, and Satoshi Hara. Characterizing the risk of fairwashing. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine Bias*. Propubilca, May, 23 2016.

Richard J Arneson. Equality and equal opportunity for welfare. In *The Notion of Equality*, pages 237–253. Routledge, 2018.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Pranjal Awasthi, Matthäus Kleindessner, and Jamie Morgenstern. Equalized odds post-processing under imperfect group information. In *International Conference on Artificial Intelligence and Statistics*, pages 1770–1780, 2020.

Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, and Aleksandra Mojsilovic et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Dua Dheeru and Efi Karra Taniskidou. *UCI machine learning repository*. 2017.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226, 2012.

Kazuto Fukuchi, Satoshi Hara, and Takanori Maehara. Faking fairness via stealthily biased sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 412–419, 2020.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015.

Gabriel Laberge, Ulrich Aïvodji, Satoshi Hara, Mario Marchand, and Khomh Fouts. Fooling SHAP with stealthily biased sampling. In *The Eleventh International Conference on Learning Representations*, 2023.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2801–2807, 2019.

Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of wasserstein distances. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Nathan Mantel and William Haenszel. Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the National Cancer Institute*, 22(4):719–748, 1959.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54 (6):1–35, 2021.

Luke Merrick and Ankur Taly. The explanation game: Explaining machine learning models using Shapley values. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 17–38. Springer, 2020.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, 2011.

Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. Mitigating bias in algorithmic hiring: Evaluating claims and practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 469–481, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*, 2018.

Clifford A Shaffer and Hanan Samet. Optimal quadtree construction algorithms. *Computer Vision, Graphics, and Image Processing*, 37(3):402–419, 1987.

Mohammad Ahmad Sheikh, Amit Kumar Goel, and Tapas Kumar. An approach for prediction of loan approval using machine learning algorithm. In *2020 International Conference on Electronics and Sustainable Communication Systems*, pages 490–494. IEEE, 2020.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

Yuki Takezawa, Ryoma Sato, Zornitsa Kozareva, Sujith Ravi, and Makoto Yamada. Fixed support tree-sliced wasserstein barycenter. pages 1120–1137, 2021.

Cédric Villani. *Optimal Transport: Old and New*. Springer, 2009.