On the Sensitivity of Individual Fairness: Measures and Robust Algorithms

Xinyu He
University of Illinois at Urbana-Champaign
USA, IL, Champaign
xhe34@illinois.edu

Jian Kang
University of Rochester
USA, NY, Rochester
jian.kang@rochester.edu

Ruizhong Qiu
University of Illinois at Urbana-Champaign
USA, IL, Champaign
rq5@illinois.edu

Fei Wang
Amazon
USA, CA, Sunnyvale
feiww@amazon.com

Jose Sepulveda
Amazon
USA, WA, Seattle
joseveda@amazon.com

Hanghang Tong
University of Illinois at Urbana-Champaign
USA, IL, Champaign
htong@illinois.edu

Abstract

Algorithmic fairness has been receiving increasing attention in recent years. Among others, individual fairness, with its root in the dictionary definition of fairness, offers a fine-grained fairness notion. At the algorithmic level, individual fairness can often be operationalized as a convex regularization term with respect to a similarity matrix. Appealing as it might be, a notorious challenge of individual fairness lies in how to find appropriate distance or similarity measure, which largely remains open to date. Consequently, the similarity or distance measure used in almost any individually fair algorithm is likely to be imperfect due to various reasons such as imprecise prior/domain knowledge, noise, or even adversaries. In this paper, we take an important step towards resolving this fundamental challenge and ask: how sensitive is the individually fair learning algorithm with respect to the given similarities? How can we make the learning results robust with respect to the imperfection of the given similarity measure? First (Soul-M), we develop a sensitivity measure to characterize how the learning outcomes of an individually fair learning algorithm change in response to the change of the given similarity measure. Second (Soul-A), based on the proposed sensitive measure, we further develop a robust individually fair algorithm by adversarial learning that optimizes the similarity matrix to defend against L_∞ attack. A unique advantage of our sensitivity measure and robust algorithm lies in that they are applicable to a broad range of learning models as long as the objective function is twice differentiable. We conduct extensive experiments to demonstrate the efficacy of our methods.

CCS Concepts

• **Information systems → Data mining**;

Keywords

Individual fairness, similarity measures, robustness, sensitivity analysis

CIKM '24, October 21–25, 2024, Boise, ID, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679721

ACM Reference Format:
Xinyu He, Jian Kang, Ruizhong Qiu, Fei Wang, Jose Sepulveda, and Hanghang Tong. 2024. On the Sensitivity of Individual Fairness: Measures and Robust Algorithms. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679721

1 Introduction

Alongside the wide deployment of machine learning algorithms in our daily lives, numerous real-world cases that suffer from unfair machine learning algorithms have come into the spotlight [30], e.g., gender discrimination in advertisement promotion systems, and racism in facial recognition systems. Therefore, developing fair machine learning algorithms has become an essential topic. Guided by the dictionary definition of fairness ("lack of favoritism toward one side or another"[1]), individual fairness has been proposed to ensure similar individuals to obtain similar learning outcomes[10]. Such fairness notion also has its root in the field of ethics and philosophy in law [12], where Aristotle's conception of justice says "like cases should be treated alike". Compared with other fairness notions such as group fairness [8, 11, 42] or counterfactual fairness [2, 13, 25], an add-on benefit of individual fairness [4, 7, 9, 29, 38, 43] lies in that it considers fairness on a finer granularity.

The seminal work by Dwork et al. [10] first operationalizes individual fairness through Lipschitz property. At its core, it bounds the similarity[2] between learning outcomes with the similarity between the input instances. Based on this overarching framework, numerous algorithms for ensuring individual fairness have been proposed since then [4, 7, 20, 29]. Among them, [20] quantifies the individual bias as a trace term with respect to the given similarity matrix. The intuition is to measure the total difference of the learning outcomes between all pairs of samples weighted by the corresponding similarity between them. By minimizing such individual bias, it naturally leads to three debiasing algorithms, namely debiasing the input data (pre-processing), debiasing the model (in-processing), and debiasing the learning outcome (post-processing). We note that

[1] https://www.merriam-webster.com/dictionary/fairness
[2] In [10], it uses distance metric for individual fairness, which can be viewed as the inverse of similarity. For clarity, we will use term similarity throughout this paper.

while [20] was originally developed for graph data, both the bias measure and debiasing algorithms therein are applicable to other types of data such as vector data.

Despite the substantial progress, several fundamental challenges still exist. First, while there exists rich similarity measures in the literature (e.g. cosine similarity, Jarccard index), it is unclear which similarity should be used for individual fairness. Moreover, with the rich literature of graph structural attack [40, 47], and given that attacks on adjacency matrix or features will affect the output similarities obtained by metrics, the robustness of similarity matrix becomes an important question to solve as well. Second, considerate efforts have been made to *learn* the similarity measure for individual fairness. For example, [26, 45] add learnable individual weight parameters for different features in the similarity measure. However, learning a weighted similarity measure from a complex objective function might be unreliable, because it is hard to tell whether the learnt similarity measure promotes individual fairness or instead penalizes individual fairness in trade of the better model utility. Another group of works including [17, 31] involves human feedback or judgements, which requires much human effort as their measurements are task-specific and human judgements themselves might be biased or malicious. Third, from the model robustness perspective, the similarity measure, as part of the input of an individually fair learning algorithm, provides the attackers an additional source to attack model's utility. It is therefore critical to understand the vulnerability of an individually fair learning model in response to such attacks.

Due to these critical challenges, how to find the appropriate similarity measure for individual fairness largely remains as an open problem to date. In this paper, we take an important step towards resolving these fundamental issues and ask: *Q1. how sensitive is the individually fair learning algorithm with respect to the given similarities, and, Q2. how can we make the learning results robust with respect to the imperfection of the given similarity measure?* First (sensitivity measure), we develop a sensitivity measure to characterize how perturbations on similarity measure impact the individually fair learning outcome, where the key idea is to efficiently estimate the gradient of the learning outcome with respect to the given similarity measure. Furthermore, we derive an approximation of our measurement using influence function that can be approximated in linear time with respect to the number of nonzero entries in the similarity matrix. Second (robust algorithm), based on the proposed sensitive measure, we further develop a robust individually fair algorithm by adversarial learning which is formulated as a nested bi-level optimization problem. The highest level optimizes the similarity to minimize the sensitivity under the worst perturbations, the middle level finds the perturbation that leads to the worst sensitivity, and the lowest level finds the best individually fair learning outcomes with respect to the given similarity measure. A unique advantage of our sensitivity measure and robust algorithm lies in that they are applicable to a broad range of learning models as long as the objective function is twice differentiable. We instantiate the proposed sensitivity measures and robust algorithms with three popular learning models, including ranking, clustering and classification. We conduct extensive experiments to demonstrate the efficacy of our methods.

Table 1: Key symbols in the paper.

Symbols	Descriptions
$\mathbf{A}, \mathbf{A}^T, \mathbf{A}^{-1}$	A matrix, its transpose and its inverse.
$\mathbf{D}_\mathbf{A}, \mathbf{L}_\mathbf{A}$	Degree and Laplacian matrix of $\mathbf{A}$, $\mathbf{L}_\mathbf{A} = \mathbf{D}_\mathbf{A} - \mathbf{A}$
$\otimes$	Kronecker Product
$\mathbf{I}_m$	An $m \times m$ identity matrix
$\mathcal{D}$	Input data (e.g., adjacency matrix, features, etc.)
θ	Set of model parameters
$\mathbf{S}$	Similarity matrix
$\widetilde{\mathbf{S}}$	Robust similarity matrix
$\mathbf{Y}$	Learning outcomes
$\mathbf{Y}^*$ or $\mathbf{Y}^*_\mathbf{S}$	Debiased learning outcomes
$\widetilde{\mathbf{Y}}^*$ or $\widetilde{\mathbf{Y}}^*_{\tilde{\mathbf{S}}}$	Robust debiased learning outcomes
$\mathbf{Y}^*_{\mathbf{S}+\Delta\mathbf{S}}$	Perturbed debiased learning outcomes
$l(\cdot)$	Loss function of learning models
$\tilde{l}(\cdot)$	Loss function of debiased learning models

To our best knowledge, this work represents the first principled effort to study the sensitivity of the individual fairness. The main contributions of the paper can be summarized as follows:

- **Sensitivity measurement.** We formally formulate the Measuring Sensitivity of Individual Fairness (Soul-M) problem and propose an effective and general solution. We instantiate our method with three popular machine learning algorithms including ranking, clustering and classification. We also address computational issue regarding how to effectively conduct such sensitivity analysis.
- **Robust algorithm.** We present Soul-A algorithm that is, to our best knowledge, the first to ensure robustness of an individually fair learning model. As a byproduct, our proposed algorithm also offers a feasible solution for optimizing the similarity measure for individual fairness.
- **Empirical Evaluations.** We conduct experiments on eight datasets. The experimental results demonstrate our methods' efficacy. For the ranking task, the proposed Soul-A algorithm reduces the sensitivity by over 99%. Our experiments also reveal that the similarity measure could effectively act as an additional source for attacking the model utility.

2 Problem Definition

In this section, we first present the key symbols used throughout the paper (Table 1). Then we briefly review individual fairness and debiasing algorithms. Finally, we formally define sensitivity measure and robust algorithm problems for individual fairness.

Notations. In this paper, we use caligraphic letters for set, bold upper-case letters for two-dimensional matrices or high-dimensional tensors, lower-case letters for vectors or hyperparameters. We follow the conventions in PyTorch for indexing. For example, $\mathbf{A}[i, :]$ denotes the i^{th} row of matrix $\mathbf{A}$.

Let $\mathcal{D}$ represent the input data of a learning model, which is the feature matrix $\mathbf{X}$ for vector data or the adjacency matrix $\mathbf{A}$ for graph.[3] For (semi-)supervised learning tasks, $\mathcal{D}$ also includes the available supervision (e.g., the class labels of the training samples). We represent the learning outcomes by a matrix $\mathbf{Y} \in \mathbb{R}^{N \times d}$, whose i^{th} row $\mathbf{Y}[i, :]$ is the learning outcome for the i^{th} sample (e.g., the embedding vector for the embedding task, the predicted class probabilities for the classification task, the cluster membership for the clustering task, the ranking score for the ranking task, etc.). $\mathbf{S}$ is a symmetric similarity matrix for the samples in the input space (i.e.,

[3] For an attributed graph, the input dataset $\mathcal{D}$ also includes the initial node feature matrix.

$S[i,j]$ measures the similarity between samples i and j in the input space). We take an optimization perspective of the learning model, where the optimal learning outcomes Y is obtained by minimizing a task-specific loss function $l(Y, \mathcal{D}, \theta)$, where θ represents the model parameters [20]. For example, $l()$ could be the cross entropy loss for the classification task or the contrastive loss for the embedding task. See Table 2 for some examples of $l()$.

Individual fairness. Individual fairness mandates to treat like cases alike. The seminar work in [10] first operationalizes individual fairness through Lipschitz property, which bounds the difference of learning outcomes between two samples (e.g., $Y[i,:]$ and $Y[j,:]$) by the difference of the corresponding samples in the input space (e.g., the inverse of $S[i,j]$). InFoRM [20] further defines individual bias of the learning outcomes Y as a trace term

$$\text{Tr}(Y^T L_S Y) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|Y[i,:] - Y[j,:]\|_2^2 S[i,j] \tag{1}$$

where $L_S = D_S - S$ is the Laplacian of the similarity matrix S and D_S is the degree matrix of S. The intuition of the bias defined in Eq. (1) is that it measures the total squared difference ($\|Y[i,:] - Y[j,:]\|_2^2$) of the learning outcomes between all pairs of samples weighted by the corresponding similarity between them ($S[i,j]$). Note that the bias defined in Eq. (1) only depends on Y and S. Therefore, even though the bias measure in [20] was originally developed for graph data, it is applicable to other types of data (such as vector data) and various learning tasks, as long as learning outcomes Y can be represented in the form of a matrix.

By minimizing the bias in Eq. (1), it naturally leads to three debiasing algorithms, namely debiasing the input data (pre-processing), debiasing the model (in-processing), and debiasing the learning outcomes (post-processing) [20]. For example, by introducing the bias term as a regularization term into the task-specific loss function $l()$, we can obtain the debiased learning outcomes Y_S^* as

$$Y_S^* = \underset{Y}{\text{argmin}}\, \tilde{l}(Y, \mathcal{D}, \theta, S) = \underset{Y}{\text{argmin}}\, l(Y, \mathcal{D}, \theta) + \alpha \text{Tr}(Y^T L_S Y) \tag{2}$$

where $\alpha > 0$ is regularization parameter that balances fairness and accuracy. Note that we use the subscript S to emphasize that the debiased learning outcomes depend on the similarity matrix S. When the context is clear, we will omit such a subscript for brevity. The trace term in Eq. (1) is convex whose partial deriviative with respect to the similarity matrix S can be computed in linear time with respect to the number of non-zero entries in S. Therefore, we can effectively solve Eq. (2) by (stochastic) gradient descent as long as the task-specific loss function $l()$ is differentiable. In this paper, we focus on developing sensitivity measure and robust algorithm for individually fair learning outcomes Y^* obtained via Eq. (2).

Problem definitions. Although Eq. (2) provides a generic solution to debias a variety of learning models, it requires the similarity measure S as its input. How to obtain the optimal similarity measure S, however, remains an open challenge. Consequently, the similarity or distance measure used in almost any individually fair algorithm is likely to be imperfect due to various reasons such as imprecise prior/domain knowledge, noise, or even adversaries. It is therefore critical to understand how the imperfection of the input similarity measure S impacts the individually-fair learning outcomes Y^*. Formally, we define the sensitivity measure problem of individually fair learning as follows.

Problem 1. Measuring Sensitivity of Individual Fairness (Soul-M).

Input: (i) Dataset $\mathcal{D}$, (ii) similarity matrix S, (iii) the individually fair learning outcomes Y_S^* obtained by Eq. (2) as well as the corresponding learning model, and (iv) symmetric perturbation on the similarity matrix ΔS (i.e., $\Delta S[i,j] = \Delta S[j,i]$);

Output: the change of the individually fair learning outcomes $\Delta Y^* \approx Y_{S+\Delta S}^* - Y_S^*$, where $Y_{S+\Delta S}^* = \text{argmin}_Y\, \tilde{l}(Y, \mathcal{D}, \theta, S + \Delta S)$.

Based on the sensitivity measure, we further seek to make the individually-fair learning outcomes robust with respect to the imperfection of the given similarity measure:

Problem 2. Robust Individually Fair Learning (Soul-A)

Input: (i) Dataset $\mathcal{D}$, (ii) similarity matrix S, (iii) the individually fair learning outcomes Y_S^* obtained by Eq. (2) as well as the corresponding learning model;

Output: (1) the revised robust similarity matrix $\widetilde{S}$, and (2) the revised individually fair learning outcomes $\widetilde{Y}_{\widetilde{S}}^* = \text{argmin}_Y\, \tilde{l}(Y, \mathcal{D}, \theta, \widetilde{S})$ that is (i) as close as possible to Y_S^*, and (ii) robust to the perturbation on the similarity measure $\widetilde{S}$.

3 Soul-M: Sensitivity Measurement

In this section, we address Problem 1 to quantify the change of individually fair learning outcomes in response to symmetric perturbation on the input similarity matrix. Mathematically, we aim to characterize the difference $\Delta Y^* \approx Y_{S+\Delta S}^* - Y_S^*$ between the individually fair learning outcomes $Y_{S+\Delta S}^*$ learned in consideration of the perturbation ΔS and its counterpart Y_S^* without symmetric perturbation. A naive solution to Problem 1 is to directly calculate ΔY by exhaustively re-training the learning model for ΔS, which is computationally expensive. To address this issue, we resort to local sensitivity analysis [5] and estimate the partial derivative $\frac{\partial Y^*}{\partial S[i,j]}$ to avoid exhaustive re-training, which quantifies how Y^* changes in response to the infinitesimal change of S (Section 3.1). Our measurement can be applied to any learning model with twice differentaible loss function $l()$. We further instantiate our general solution with three representative learning models, including PageRank, spectral clustering, and binary classification (Section 3.2).

3.1 A General Solution

Following the overarching principle of local sensitivity analysis [5], we need to compute two key terms to estimate the sensitivity of individually fair learning outcomes with respect to a symmetric perturbation ΔS: (1) the partial derivative $\frac{\partial Y^*}{\partial S[i,j]}$, and (2) the product between the partial derivative $\frac{\partial Y^*}{\partial S[i,j]}$ and the symmetric perturbation ΔS. We first provide the following theorem (Theorem 3.1) for the computation of partial derivative. Building upon our results in Theorem 3.1, we present our proposed sensitivity measurement in Theorem 3.2. Finally, we introduce Algorithm 1 for efficient computation of the sensitivity measurement.

Computing the partial derivative $\frac{\partial Y^*}{\partial S[i,j]}$. Following influence function [24], we compute the partial derivative by considering the learning outcomes as the parameters and an infinitesimal perturbation on similarity matrix as upweighting the loss function. Its rationale is that the individual bias (Eq. (1)) is a weighted sum over the differences of learning outcomes between two samples with similarity as the weights. However, directly applying influence function [24] requires us to compute a second-order derivative $\frac{1}{n}\left(\frac{\partial^2 \tilde{l}(Y, \mathcal{D}, \theta, S)}{\partial Y^2}\right)$

of the debiased loss function $\tilde{l}(\mathrm{Y}, \mathcal{D}, \theta, \mathrm{S})$ with respect to the learning outcomes Y, which could naturally be a high-dimensional tensor if Y is a 2-dimensional matrix (e.g., an embedding matrix, predicted probability matrix of a multi-class classification problem). To avoid the non-trivial tensor operations, we observe that each element in the tensor is equivalent to computing the second-order derivative with respect to each element in the first-order derivative matrix, i.e., $\left(\frac{\partial^2 \tilde{l}(\mathrm{Y},\mathcal{D},\theta,\mathrm{S})}{\partial \mathrm{Y}^2}\right)[i,j,k,l] = \frac{\partial^2 \tilde{l}(\mathrm{Y},\mathcal{D},\theta,\mathrm{S})}{\partial \mathrm{Y}[i,j]\partial \mathrm{Y}[k,l]}$. Thus, similar to [23], we construct an $(Nd) \times (Nd)$ matrix $\mathbf{W}$, namely Soul-M matrix, by computing the second-order derivative $\frac{\partial^2 \tilde{l}(\mathrm{Y},\mathcal{D},\theta,\mathrm{S})}{\partial \operatorname{vec}(\mathrm{Y})^2}$ of the loss function with respect to the vectorized learning outcomes $\operatorname{vec}(\mathrm{Y})$. With $\operatorname{vec}(\mathrm{Y})$, we essentially compute the $\frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]}$ which is equivalent to vectorizing the partial derivative $\frac{\partial \mathrm{Y}^*}{\partial \mathrm{S}[i,j]}$. Theorem 3.1 presents how to construct the Soul-M matrix $\mathbf{W}$ and compute the vectorized partial derivative $\frac{\partial \mathrm{Y}^*}{\partial \mathrm{S}[i,j]}$.

Theorem 3.1. *Define* $\mathbf{W} = \left(\frac{\partial^2 l(\mathrm{Y},\mathcal{D},\theta)}{\partial \operatorname{vec}(\mathrm{Y})^2} + 2\alpha \mathbf{L}_{\mathrm{S}} \otimes \mathbf{I}_d\right)^{-1} \in \mathbb{R}^{Nd \times Nd}$ *as Soul-M matrix, where* $\otimes$ *is the Kronecker product,* $\operatorname{vec}()$ *vectorizes a matrix into a column vector. Denote the debiased learning outcomes obtained from Eq.* (2) *as* Y^*, *and assume the task-specific loss function* $l()$ *is twice differentiable and strictly convex.* $\forall\ i, j \in \{1, \ldots, N\}$ *we have that the partial derivative of* Y^* *over* $\mathrm{S}[i,j]$ *is*

$$\frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]} = -\alpha \sum_{t=0}^{d-1} (\mathbf{W}[:, tN+i] - \mathbf{W}[:, tN+j]) \cdot \\ (\operatorname{vec}(\mathrm{Y}^*)[tN+i] - \operatorname{vec}(\mathrm{Y}^*)[tN+j]) \tag{3}$$

Proof. Applying influence function in [24] to Eq. (2) and vectorize Y^*, we get

$$\begin{aligned}\frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]} &= -\frac{1}{n}\mathbf{H}_{\mathrm{Y}}^{-1} \frac{\partial^2 (l(\mathrm{Y},\mathcal{D},\theta) + \alpha \operatorname{Tr}(\operatorname{vec}(\mathrm{Y})^T (\mathbf{L}_{\mathrm{S}} \otimes \mathbf{I}_d) \operatorname{vec}(\mathrm{Y})))}{\partial \mathrm{S}[i,j] \partial \operatorname{vec}(\mathrm{Y})} \\ \mathbf{H}_{\mathrm{Y}} &= \frac{1}{n}\left(\frac{\partial^2 (l(\mathrm{Y},\mathcal{D},\theta) + \alpha \operatorname{Tr}(\operatorname{vec}(\mathrm{Y})^T (\mathbf{L}_{\mathrm{S}} \otimes \mathbf{I}_d) \operatorname{vec}(\mathrm{Y})))}{\partial \operatorname{vec}(\mathrm{Y})^2}\right) \\ &= \frac{1}{n}\left(\frac{\partial^2 l(\mathrm{Y},\mathcal{D},\theta)}{\partial \operatorname{vec}(\mathrm{Y})^2} + 2\alpha \mathbf{L}_{\mathrm{S}} \otimes \mathbf{I}_d\right)\end{aligned} \tag{4}$$

where n is the number of nonzero entries in S and $\mathbf{H}_{\mathrm{Y}}$ is Hessian matrix. Since S is only used for calculating individual fairness bias and is irrelevant to $l(\mathrm{Y},\mathcal{D},\theta)$, the partial derivative of $l(\mathrm{Y},\mathcal{D},\theta)$ over $\mathrm{S}[i,j]$ is 0,

$$\frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]} = -\frac{1}{n}\mathbf{H}_{\mathrm{Y}}^{-1} 2\alpha\left(\frac{\partial \mathbf{L}_{\mathrm{S}}}{\partial \mathrm{S}[i,j]} \otimes \mathbf{I}_d\right) \operatorname{vec}(\mathrm{Y}) \tag{5}$$

As $\mathbf{L}_{\mathrm{S}}$ is a symmetric matrix derived from $\mathbf{L}_{\mathrm{S}} = \mathbf{D}_{\mathrm{S}} - \mathrm{S}$, where $\mathbf{D}_{\mathrm{S}}$ is defined as $\mathbf{D}_{\mathrm{S}}[i,i] = \sum_{j=1}^{N} \mathrm{S}[i,j]$ and $\mathbf{D}_{\mathrm{S}}[i,j] = 0 (i \neq j)$ with N as the number of inputs, we can directly derive the gradient of $\mathbf{L}_{\mathrm{S}}$ with respect to S and get

$$\frac{\partial \operatorname{vec}(\mathrm{Y})}{\partial \mathrm{S}[i,j]} = -\frac{\alpha}{n} \sum_{t=0}^{d-1} (\mathbf{H}_{\mathrm{Y}}^{-1}[:, tN+i] - \mathbf{H}_{\mathrm{Y}}^{-1}[:, tN+j]) \cdot \\ (\operatorname{vec}(\mathrm{Y}^*)[tN+i] - \operatorname{vec}(\mathrm{Y}^*)[tN+j]) \tag{6}$$

With Eq. (4), rewrite Eq. (6) as Eq. (3) which completes the proof. □

Computing the sensitivity measurement $\Delta\mathrm{Y}^*$. Theorem 3.1 characterizes the rate of the change of learning outcomes Y^* with respect to the infinitesimal change in the similarity $\mathrm{S}[i,j]$ of two input instances i and j. However, to compute $\Delta\mathrm{Y}^*$, it is likely that perturbation $\Delta\mathrm{S}$ on the similarity matrix S is not infinitesimal. Thus, to respond to the non-infinitesimal change $\Delta\mathrm{S}$, we assume any $\mathrm{S}[i,j]$, $\forall i,j \in \{1,\ldots,N\}$ are independent in the training process to ignore the mutual influence among different $\Delta\mathrm{S}[i,j]$ on learning outcomes and compute $\operatorname{vec}(\Delta\mathrm{Y}^*) \approx \sum_i \sum_j \frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]} \Delta\mathrm{S}[i,j]$. Theorem 3.2 summarizes computation of $\Delta\mathrm{Y}^*$.

Theorem 3.2. *Let* $\mathbf{W} = \left(\frac{\partial^2 l(\mathrm{Y},\mathcal{D},\theta)}{\partial \operatorname{vec}(\mathrm{Y})^2} + 2\alpha \mathbf{L}_{\mathrm{S}} \otimes \mathbf{I}_d\right)^{-1}$ *be Soul-M matrix. Given debiased learning outcomes* Y^* *obtained from Eq.* (2) *and symmetric perturbation on similarity matrix* $\boldsymbol{\Delta}\mathbf{S}$. *If the task-specific loss function* $l()$ *is twice differentiable and strictly convex, the difference between learning outcomes with or without perturbation* $\Delta\mathrm{S}$ *can be approximated by*

$$\operatorname{vec}(\Delta\mathrm{Y}^*) = -2\alpha \mathbf{W}(\mathbf{L}_{\Delta\mathrm{S}} \otimes \mathbf{I}_d) \operatorname{vec}(\mathrm{Y}^*) + O(\|\Delta\mathrm{S}\|^2) \tag{7}$$

Proof. Estimating $\operatorname{vec}(\Delta\mathrm{Y}^*)$ with its first order Taylor expansion over S and plugging in Theorem 3.1,

$$\begin{aligned}\operatorname{vec}(\Delta\mathrm{Y}^*) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial \operatorname{vec}(\mathrm{Y}^*)}{\partial \mathrm{S}[i,j]} \Delta\mathrm{S}[i,j] + O(\|\Delta\mathrm{S}\|^2) \\ &= -\alpha \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=0}^{d-1} (\mathbf{W}[:, tN+i] - \mathbf{W}[:, tN+j]) \cdot \\ &\qquad (\operatorname{vec}(\mathrm{Y}^*)[tN+i] - \operatorname{vec}(\mathrm{Y}^*)[tN+j]) \Delta\mathrm{S}[i,j] \\ &\quad + O(\|\Delta\mathrm{S}\|^2) \\ &= -\alpha \sum_{i=1}^{Nd}\sum_{j=1}^{Nd} (\mathbf{W}[:, i] - \mathbf{W}[:, j]) \cdot \\ &\qquad (\operatorname{vec}(\mathrm{Y}^*)[i] - \operatorname{vec}(\mathrm{Y}^*)[j]) (\Delta\mathrm{S} \otimes \mathbf{I}_d)[i,j] \\ &\quad + O(\|\Delta\mathrm{S}\|^2)\end{aligned} \tag{8}$$

Note that the second equality holds because $\forall |i-j| \geq N$, $(\Delta\mathrm{S} \otimes \mathbf{I}_d)[i,j] = 0$. Rearrange Eq. (8), we get Eq. (7) completes the proof. □

As the second order error term in Theorem 3.2 is small enough to be ignored, we can measure the sensitivity by a product of inverse of Hessian matrix (Soul-M matrix $\mathbf{W}$) and a vector $((\mathbf{L}_{\Delta\mathrm{S}} \otimes \mathbf{I}_d) \operatorname{vec}(\mathrm{Y}^*))$ of length Nd. Then we further apply a scaled Hessian-vector product [1, 24] to reduce the $O(N^3 d^3)$ complexity in computing the matrix inversion to linear complexity. The workflow of the scaled Hessian-vector product is presented in Algorithm 1. In detail, with proper initialization of $\mathbf{H} = c\mathbf{W}^{-1}$ and $\mathbf{v}$ (lines 1 – 2), we recursively derive the i^{th} order approximation of Hessian-vector product $\mathbf{H}_{\mathbf{v}}^{(i)}$ by

$$\mathbf{H}_{\mathbf{v}}^{(i)} = \mathbf{v} + (\mathbf{I} - \mathbf{H})\mathbf{H}_{\mathbf{v}}^{(i-1)}, i = \{1, ..., k\}, \mathbf{H}_{\mathbf{v}}^{(0)} = \mathbf{v} \tag{9}$$

(lines 3 – 6). After the Hessian-vector product is obtained, we compute the change of debiased learning outcomes (lines 7 – 8). Comparing to Hessian-vector product, Algorithm 1 introduces a scaling factor c during initialization (line 1) and undo the scaling later (line 7). The rationale of the scaling factor is as follows. Hessian-vector product is a power iteration-based method that computes the Taylor expansion iteratively $\mathbf{B}^{-1} = \sum_{j=0}^{\infty} (\mathbf{I} - \mathbf{B})^j$ for a matrix $\mathbf{B}$. To guarantee convergence, the series $\{(\mathbf{I} - \mathbf{B})^j\}$ should converges to zero, which requires the magnitude of leading eigenvalue of $(\mathbf{I} - \mathbf{B})$ to be upper bounded by 1. Thus, the scaling factor c is introduced such that the magnitude of leading eigenvalue of $\mathbf{I} - c\mathbf{W}^{-1}$ is upper bounded by 1, equivalent to requiring the

Algorithm 1: Scaled Hessian-vector product to estimate $\text{vec}(\Delta\mathbf{Y}^*)$

```
Input  : (i) Second-order derivative $\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\,\text{vec}(\mathbf{Y})^2}$, (ii)
         regularization parameter $\alpha$, (iii) Laplacian of
         similarity matrix and perturbation $\mathbf{L_S}, \mathbf{L_{\Delta S}}$, (iv)
         debiased learning outcomes $\mathbf{Y}^*$, and (v) the scaling
         factor $c$
Output : Change of debiased learning outcomes before and
         after perturbation $\text{vec}(\Delta\mathbf{Y}^*)$
Initialize Hessian matrix with scaling
  $\mathbf{H} \leftarrow c\left(\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\,\text{vec}(\mathbf{Y})^2} + 2\alpha\mathbf{L_S}\otimes\mathbf{I}_d\right)$;
Initialize $\mathbf{v} \leftarrow (\mathbf{L_{\Delta S}}\otimes\mathbf{I}_d)\,\text{vec}(\mathbf{Y}^*)$;
Initialize Hessian-vector product $\mathbf{H}_\mathbf{v}^{(0)} \leftarrow \mathbf{v}$;
for $i$ in $[1:k]$ do
    Update $\mathbf{H}_\mathbf{v}^{(i)} \leftarrow \mathbf{v} + (\mathbf{I}-\mathbf{H})\mathbf{H}_\mathbf{v}^{(i-1)}$;
end
Undo scaling $\mathbf{H}_\mathbf{v} \leftarrow c\mathbf{H}_\mathbf{v}^{(k)}$;
return $\text{vec}(\Delta\mathbf{Y}^*) \leftarrow -2\alpha\mathbf{H}_\mathbf{v}$;
```

magnitude of leading eigenvalue of $c\mathbf{W}^{-1}$ being upper bounded by 2. Lemma 3.3 demonstrates the existence of scaling factor c.

Lemma 3.3. *For any scaling factor c satisfying*

$$0 < c < \frac{2}{\|(\mathbf{W}^{-1})^T\|_1} \tag{10}$$

we have that

$$0 < |\lambda^{(i)}_{c\mathbf{W}^{-1}}| < 2 \quad \forall i \tag{11}$$

where $\|(\mathbf{W}^{-1})^T\|_1 = \max_j(\sum_t |\mathbf{W}^{-1}[j,t]|)$ is the matrix 1-norm, and $\lambda^{(i)}_{c\mathbf{W}^{-1}}$ is the i^{th} eigenvalue of $c\mathbf{W}^{-1}$.

Proof. The lower bound of absolute eigenvalue is guaranteed as $\mathbf{W}^{-1} = n\mathbf{H}_\mathbf{Y}$, whereas eigenvalues of Hessian matrices, which denotes the convexity of loss function, is positive if loss function is strictly convex and ideally optimized.

For the upper bound of absolute eigenvalue, according to Gershgorin circle theorem [41],

$$\forall i,\ |\lambda^{(i)}_{\mathbf{W}^{-1}}| \le \max_j(|\mathbf{W}^{-1}[j,j]| + \sum_{t\neq j}|\mathbf{W}^{-1}[j,t]|) \tag{12}$$

Therefore, $|\lambda^{(i)}_{c\mathbf{W}^{-1}}| = c|\lambda^{(i)}_{\mathbf{W}^{-1}}| < 2$ is ensured if Eq. (10) holds. □

Lemma 3.3 shows that the scaling factor c that satisfies the convergence guarantee is upper bounded by $\frac{2}{\|(\mathbf{W}^{-1})^T\|_1}$. In practice, we initialize $c = \frac{1}{\|(\mathbf{W}^{-1})^T\|_1}$, i.e., by taking the reciprocal with respect to the matrix 1-norm of the transpose of Hessian matrix $\mathbf{W}^{-1}$. In summary, the time complexity of Algorithm 1 is as follows.

Remark 1. *If $l(\mathbf{Y},\mathcal{D},\theta)$ satisfies $\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\mathbf{Y}[i,:]\partial\mathbf{Y}[j,:]} = \mathbf{0}, \forall i \neq j$, the time complexity of Algorithm 1 is $O(kMd)$, where k is the number of iterations (the order of Taylor expansion), M is the number of nonzero entries in $\mathbf{L_S}$, and d is the dimension of the learning outcome for each input instance.*

Note that the assumption $\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\mathbf{Y}[i,:]\partial\mathbf{Y}[j,:]} = \mathbf{0}, \forall i \neq j$ is mild and holds for many commonly used objectives such as (binary) cross-entropy loss and mean squared error.

3.2 Soul-M: Instantiations

The key to compute the sensitivity measure $\Delta\mathbf{Y}^*$ is to compute the second-order derivative $\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\,\text{vec}(\mathbf{Y})^2}$. In this section, we present three instantiations on the computation of $\frac{\partial^2 l(\mathbf{Y},\mathcal{D},\theta)}{\partial\,\text{vec}(\mathbf{Y})^2}$ as well as the Soul-M matrix $\mathbf{W}$ for different learning models, including PageRank [33], spectral clustering [32], and binary cross-entropy loss optimization (e.g., logistic regression, neural networks). We summarize three instantiations as well as the corresponding second-order derivative and Soul-M matrix in Table 2.

PageRank [33] is a graph ranking algorithms based on random walk. It outputs a column ranking vector $\mathbf{Y}^*$ by recursively computing the following linear system until convergence

$$\mathbf{Y} = p\mathbf{A}\mathbf{Y} + (1-p)\mathbf{e} \tag{13}$$

where $\mathbf{Y}$ is a 1-dimensional column vector of ranking scores, $0 < p < 1$ is the damping factor, and $\mathbf{A}$ is the normalized transition matrix, and $\mathbf{e}$ is a pre-defined teleportation vector. Mathematically, PageRank is equivalent to minimize the following objective function when $\mathbf{A}$ is symmetrically normalized.

$$\mathbf{Y}^* = \underset{\mathbf{Y}}{\text{argmin}}\ \frac{1}{2}(p\mathbf{Y}^T(\mathbf{I}-\mathbf{A})\mathbf{Y} + (1-p)\|\mathbf{Y}-\mathbf{e}\|_F^2) \tag{14}$$

Regarding the second-order derivative, since $\mathbf{Y}$ is a column vector (i.e., $d=1$), we have $\mathbf{Y} = \text{vec}(\mathbf{Y})$ and $\mathbf{I}_d = 1$. Then by the properties of the derivative of matrix multiplication and the Frobenius norm, we have that

$$\frac{\partial^2 \frac{1}{2}(p\mathbf{Y}^T(\mathbf{I}-\mathbf{A})\mathbf{Y} + (1-p)\|\mathbf{Y}-\mathbf{e}\|_F^2)}{\partial\,\text{vec}(\mathbf{Y})^2} = \mathbf{I} - p\mathbf{A} \tag{15}$$

Combining Eq. (15) and $\mathbf{L_S}\otimes\mathbf{I}_d$, we have that

$$\mathbf{W} = (\mathbf{I} - p\mathbf{A} + \alpha\mathbf{L_S}\times 1)^{-1} = (\mathbf{I} - p\mathbf{A} + \alpha\mathbf{L_S})^{-1} \tag{16}$$

Spectral clustering [32] is a clustering algorithm that group nodes in a graph $\mathbf{A}$ into d clusters by their spectral embeddings. Specifically, the spectral embedding matrix $\mathbf{Y}^*$ of all nodes in the graph is equivalent to a matrix whose columns are the eigenvectors of graph Laplacian $\mathbf{L_A}$ corresponding to d smallest eigenvalues. Mathematically, it solves the following optimization problem.

$$\mathbf{Y}^* = \underset{\mathbf{Y}}{\text{argmin}}\ \text{Tr}(\mathbf{Y}^T\mathbf{L_A}\mathbf{Y}) \tag{17}$$

By the properties of derivative of matrix multiplication and the trace operator, we have that

$$\frac{\partial^2\text{Tr}(\mathbf{Y}^T\mathbf{L_A}\mathbf{Y})}{\partial\,\text{vec}(\mathbf{Y})^2} = \mathbf{L_{2A}}\otimes\mathbf{I}_d \tag{18}$$

Then we have

$$\frac{\partial^2\text{Tr}(\mathbf{Y}^T\mathbf{L_A}\mathbf{Y})}{\partial\,\text{vec}(\mathbf{Y})^2} + 2\alpha\mathbf{L_S}\otimes\mathbf{I}_d = \mathbf{L_{2A+2\alpha S}}\otimes\mathbf{I}_d \tag{19}$$

However, the Laplacian matrix $\mathbf{L_{2A+2\alpha S}}$ is singular. Consequently, the second-order derivative is also singular $\mathbf{L_{2A+2\alpha S}}\otimes\mathbf{I}_d$, which does not have the corresponding inverse matrix. Thus, we approximate Soul-M matrix as the pseudo-inverse of $\mathbf{L_{2A+2\alpha S}}\otimes\mathbf{I}_d$ and get

$$\mathbf{W} = (\mathbf{L_{2A+2\alpha S}}\otimes\mathbf{I}_d)^\dagger \tag{20}$$

Binary cross-entropy loss is a commonly used loss function to optimize learning models like logistic regression or neural networks for classification. It is mathematically defined as

$$l = -\frac{1}{N}\sum_{i=1}^{N}(\mathbf{Y}_{\text{gt}}[i]\log(\mathbf{Y}[i]) + (1-\mathbf{Y}_{\text{gt}}[i])\log(1-\mathbf{Y}[i])) \tag{21}$$

where N is the number of training instances, $\mathbf{Y}_{\text{gt}}[i] \in \{0,1\}$ is the ground-truth label for the i^{th} instance, and $\mathbf{Y}[i]$ is the predicted probabilities of the i^{th} instance being classified with label 1. To

Table 2: Soul-M instantiations. Note that Algorithm 1 does not need to compute the Soul-M matrix.

Mining task	Loss function l	Second-order derivative $\frac{\partial^2 l(\mathrm{Y},\mathcal{D},\theta)}{\partial \operatorname{vec}(\mathrm{Y})^2}$	Soul-M matrix $\mathbf{W}$
PageRank	$\frac{1}{2}(p\mathrm{Y}^T(\mathbf{I}-\mathrm{A})\mathrm{Y}+(1-p)\lVert \mathrm{Y}-\mathbf{e}\rVert_F^2)$	$\mathbf{I}-p\mathrm{A}$	$(\mathbf{I}-p\mathrm{A}+\alpha\mathrm{L}_\mathrm{S})^{-1}$
Spectral clustering	$\mathrm{Tr}(\mathrm{Y}^T\mathbf{L}_\mathbf{A}\mathrm{Y})$	$\mathbf{L}_{2\mathrm{A}}\otimes\mathbf{I}_d$	$(\mathbf{L}_{2\mathrm{A}+2\alpha\mathrm{S}}\otimes\mathbf{I}_d)^\dagger$
Binary cross-entropy loss	$-\frac{1}{N}\sum_{i=1}^N(\mathrm{Y}_{\mathrm{gt}}[i]\log(\mathrm{Y}[i])+(1-\mathrm{Y}_{\mathrm{gt}}[i])\log(1-\mathrm{Y}[i]))$	$\left(\frac{\partial^2 l}{\partial\operatorname{vec}(\mathrm{Y})^2}\right)[i,j]=\begin{cases}\frac{1}{N}\frac{\mathrm{Y}[i]^2-2\mathrm{Y}[i]\mathrm{Y}_{\mathrm{gt}}[i]+\mathrm{Y}_{\mathrm{gt}}[i]}{\mathrm{Y}[i]^2(1-\mathrm{Y}[i])^2}, & i=j\\ 0, & i\neq j\end{cases}$	$\left(\frac{\partial^2 l}{\partial\operatorname{vec}(\mathrm{Y})^2}+\frac{2\alpha}{N}\mathrm{L}_\mathrm{S}\right)^{-1}$

align with the empirical binary cross-entropy loss, we compute the average individual bias for all training instances and solve the following regularized optimization problem.

$$\mathrm{Y}^* = \underset{\mathrm{Y}}{\operatorname{argmin}} -\frac{1}{N}\sum_{i=1}^{N}(\mathrm{Y}_{\mathrm{gt}}[i]\log(\mathrm{Y}[i])+(1-\mathrm{Y}_{\mathrm{gt}}[i])\log(1-\mathrm{Y}[i])) + \frac{\alpha}{N}\mathrm{Tr}(\mathrm{Y}^T\mathrm{L}_\mathrm{S}\mathrm{Y})$$

Because the learning outcomes Y is a column vector of predicted probabilities, similar to our instantiation with PageRank, we have $\mathrm{Y}=\operatorname{vec}(\mathrm{Y})$ and $\mathbf{I}_d=1$. Then, for the second-order derivative, we have that

$$\frac{\partial^2 l}{\partial\mathrm{Y}[i]\partial\mathrm{Y}[j]} = \frac{\partial}{\partial\mathrm{Y}[j]}\left(-\frac{1}{N}\left(\frac{\mathrm{Y}_{\mathrm{gt}}[i]}{\mathrm{Y}[i]}-\frac{1-\mathrm{Y}_{\mathrm{gt}}[i]}{1-\mathrm{Y}[i]}\right)\right) \tag{22}$$

When $i=j$, it is trivial that $\frac{\partial^2 l}{\partial\mathrm{Y}[i]\partial\mathrm{Y}[i]}=\frac{1}{N}\frac{\mathrm{Y}[i]^2-2\mathrm{Y}[i]\mathrm{Y}_{\mathrm{gt}}[i]+\mathrm{Y}_{\mathrm{gt}}[i]}{\mathrm{Y}[i]^2(1-\mathrm{Y}[i])^2}$. Otherwise, because $\mathrm{Y}[i]$ and $\mathrm{Y}[j]$ are independent, resulting in $\frac{\partial l}{\partial\mathrm{Y}[i]\partial\mathrm{Y}[i]}=0$. Putting everything together, we have

$$\mathbf{W}=\left(\frac{1}{N}\left(\frac{\partial^2 l}{\partial\mathrm{Y}^2}-2\alpha\mathrm{L}_\mathrm{S}\right)\right)^{-1} \tag{23}$$

where $\frac{\partial^2 l}{\partial\mathrm{Y}^2}$ is a diagonal matrix such that

$$\frac{\partial^2 l}{\partial\mathrm{Y}^2}[i,j]=\begin{cases}\frac{1}{N}\frac{\mathrm{Y}[i]^2-2\mathrm{Y}[i]\mathrm{Y}_{\mathrm{gt}}[i]+\mathrm{Y}_{\mathrm{gt}}[i]}{\mathrm{Y}[i]^2(1-\mathrm{Y}[i])^2}, & i=j\\ 0, & i\neq j\end{cases} \tag{24}$$

4 Soul-A: Robust Algorithm

Existing works on individual fairness could be vulnerable to attacks on similarity measure. On the one hand, some works with human judgement or feedback involved can be attacked by malicious users. On the other hand, the similarities that are derived from similarity metrics can also be affected by graph structural attacks. Building upon our sensitivity measurement Soul-M, we first discuss how to attack the fair learning algorithm by manipulating the similarity measure S and then propose a method called Soul-A to improve the worst-case robustness of the fair learning algorithm against the similarity measure S.

4.1 Attacking the Fair Learning Algorithm

To quantify the worst-case change in the outcome, we consider adversarially attacking the individually fair learning algorithm by manipulating the similarity matrix S. Since the similarity matrix S is often sparse, we only consider sparse perturbations ΔS. That is, the perturbation ΔS is chosen from the set $\mathcal{S}_\mathcal{I} := \{\Delta\mathrm{S}\in\mathbb{R}^{N\times N} : \Delta\mathrm{S}[i,j]=0, \forall(i,j)\notin\mathcal{I}\}$, where $\mathcal{I}:=\{(i,j):\mathrm{S}[i,j]\neq 0\}$ denotes the index set of nonzero entries in S.

The objective of the adversarial attack is to maximize the distortion in the outcome Y^*. We aim to define the distortion appropriately so that it can be computed efficiently. Recall that Algorithm 1 can efficiently compute the approximate change $\operatorname{vec}(\Delta\mathrm{Y}^*(\mathrm{S},\Delta\mathrm{S})) := -2\alpha\mathbf{W}_\mathrm{S}(\mathbf{L}_{\Delta\mathrm{S}}\otimes\mathbf{I}_d)\operatorname{vec}(\mathrm{Y}_\mathrm{S}^*)$ in the outcome, where Y_S^* and $\mathbf{W}_\mathrm{S}$ denote the outcome and the Soul-M matrix w.r.t. the similarity matrix S, respectively. Thus, we define *approximate distortion* as:

$$J(\mathrm{S},\Delta\mathrm{S}) := \lVert\Delta\mathrm{Y}^*(\mathrm{S},\Delta\mathrm{S})\rVert_\mathrm{F} = \lVert 2\alpha\mathbf{W}_\mathrm{S}(\mathbf{L}_{\Delta\mathrm{S}}\otimes\mathbf{I}_d)\operatorname{vec}(\mathrm{Y}_\mathrm{S}^*)\rVert_\mathrm{F}. \tag{25}$$

Then, we find the optimal perturbation ΔS for the attack by maximizing the approximate distortion $J(\Delta\mathrm{S})$ under the constraint that any $|\Delta\mathrm{S}[i,j]|$ is at most ϵ for a given small allowance $\epsilon>0$:

$$\max_{\Delta\mathrm{S}\in\mathcal{S}_\mathcal{I}} J(\mathrm{S},\Delta\mathrm{S}), \text{ s.t. } \lVert\Delta\mathrm{S}\rVert_\infty\leq\epsilon. \tag{26}$$

To handle the constraint $\lVert\Delta\mathrm{S}\rVert_\infty\leq\epsilon$, we use a reparameterization $\Delta\mathrm{S}:=\epsilon\tanh\mathrm{Q}$ for a matrix $\mathrm{Q}\in\mathcal{S}_\mathcal{I}$. Since the range of $\epsilon\tanh\mathrm{Q}$ is $(-\epsilon,\epsilon)$, the constraint $\lVert\Delta\mathrm{S}\rVert_\infty\leq\epsilon$ is naturally satisfied. This enables us to apply the unconstrained gradient descent w.r.t. Q.

4.2 Robust Fair Learning

Now we introduce our Soul-A algorithm to robustify the individually fair learning algorithm against the similarity matrix S. The key idea of our Soul-A is to learn a new similarity matrix $\widetilde{\mathrm{S}}$ such that the individually fair learning outcomes $\mathrm{Y}_{\widetilde{\mathrm{S}}}^*$ w.r.t. the new $\widetilde{\mathrm{S}}$ is robust to the adversarial attack proposed in Section 4.1. Besides that, since we need the new similarity matrix $\widetilde{\mathrm{S}}$ to preserve as much information as possible in the original similarity matrix S, we consider an additional constraint $\lVert\widetilde{\mathrm{S}}-\mathrm{S}\rVert_\infty\leq\gamma$ for some $\gamma>0$. Building upon the intuition above, we propose to find the new $\widetilde{\mathrm{S}}$ via adversarial training:

$$\min_{\widetilde{\mathrm{S}}\in\mathcal{S}_\mathcal{I}}\max_{\Delta\mathrm{S}\in\mathcal{S}_\mathcal{I}} J(\widetilde{\mathrm{S}},\Delta\mathrm{S}), \quad \text{s.t. } \lVert\Delta\mathrm{S}\rVert_\infty\leq\epsilon, \quad \lVert\widetilde{\mathrm{S}}-\mathrm{S}\rVert_\infty\leq\gamma. \tag{27}$$

Eq. (27) is a nested bi-level optimization problem, as $J(\widetilde{\mathrm{S}},\Delta\mathrm{S})$ depends on $\mathrm{Y}_{\widetilde{\mathrm{S}}}^* = \operatorname{argmin}_\mathrm{Y}\tilde{l}(\mathrm{Y},\mathcal{D},\theta,\widetilde{\mathrm{S}})$. We propose to solve this nested bi-level optimization via alternating gradient descent, which is presented in Algorithm 2. Similar to Section 4.1, we use the reparameterization $\Delta\mathrm{S}:=\epsilon\tanh\mathrm{Q}$ to handle the constraint $\lVert\Delta\mathrm{S}\rVert_\infty\leq\epsilon$. Meanwhile, we do not explicitly enforce the constraint $\lVert\widetilde{\mathrm{S}}-\mathrm{S}\rVert_\infty\leq\gamma$. Instead, we control the difference between $\widetilde{\mathrm{S}}$ and S via a limited number of training iterations of $\widetilde{\mathrm{S}}$. Besides, We indirectly update S by updating its Laplacian with an equivalent effect. We remark that we do not limit the number of iterations of $\mathrm{Y}_{\widetilde{\mathrm{S}}}^*$ and ΔS and allow them to converge.

In detail, with proper initialization (lines 1 – 3), we conduct adversarial training for n epochs. In each epoch, we find $\widetilde{\mathrm{S}}$ which is robust to ΔS using gradient descent (lines 5 – 8). Then, we update the individually fair learning outcomes and find **ΔS** that leads to the highest distortion w.r.t the new $\widetilde{\mathrm{S}}$ (lines 9 – 15). After training is completed, we output the robust debiased learning outcomes.

5 Experiments

In this section, we present our experimental evaluations. The experiments are designed to answer the following questions, including

- **Q1:** (*Sensitivity Measure*) How accurately can Soul-M measurement help to estimate the change of fair learning outcomes on real-world datasets when input similarity measure is perturbed?
- **Q2:** (*Robust Algorithm*) How robust is Soul-A against noises of the input similarity measure? How can we use the proposed Soul-A to attack an individually fair learning model?

Algorithm 2: Soul-A Algorithm

```
Input   : (i) number of overall training epochs $n$; (ii)
          numbers of training iterations $n_S$ and $n_M$ for
          updating $\widetilde{\mathrm{S}}$ and $\Delta\mathrm{S}$, respectively, in one epoch;
          (iii) individually fair learning model
          $\tilde{l}(\mathrm{Y}, \mathcal{D}, \theta, \mathrm{S})$; and (iv) perturbation allowance
          $\epsilon$;
Output  : Robust individually fair learning outcomes $\widetilde{\mathbf{Y}}^*_{\widetilde{\mathrm{S}}}$
Randomly initialize $\mathbf{Q}$;
Initialize $\Delta\mathrm{S} \leftarrow \epsilon \tanh \mathbf{Q}$, $\mathbf{L}_{\widetilde{\mathrm{S}}} \leftarrow \mathbf{D}_{\mathrm{S}} - \mathrm{S}$;
Initialize $\widetilde{\mathbf{Y}}^*_{\widetilde{\mathrm{S}}} \leftarrow \mathrm{argmin}_Y\, \tilde{l}(\mathrm{Y}, \mathcal{D}, \theta, \mathrm{S})$ by Eq. (2);
for $i$ in $[1 : n]$ do
    for $j$ in $[1 : n_S]$ do
        Calculate $J(\widetilde{\mathrm{S}}, \Delta\mathrm{S})$ by Eq. (26);
        Update $\mathbf{L}_{\widetilde{\mathrm{S}}}$ via gradient descent;
    end
    for $j$ in $[1 : n_M]$ do
        Calculate $\tilde{l}(\widetilde{\mathbf{Y}}^*_{\widetilde{\mathrm{S}}}, \mathcal{D}, \theta, \widetilde{\mathrm{S}})$ by Eq. (2);
        Update $\widetilde{\mathbf{Y}}^*_{\widetilde{\mathrm{S}}}$ via gradient descent;
        Calculate $J(\widetilde{\mathrm{S}}, \Delta\mathrm{S})$ by Eq. (26);
        Update $\mathbf{Q}$ via gradient ascent;
        Update $\Delta S \leftarrow \epsilon \tanh \mathbf{Q}$;
    end
end
return robust debiased learning outcomes $\widetilde{\mathbf{Y}}^*_{\widetilde{\mathrm{S}}}$
```

5.1 Experimental Settings

The source code will be released upon the publication of the paper.

Datasets. Statistics of the datasets are summarized in Table 3.

Table 3: Statistics of datasets.

Data type	Dataset	Nodes/Instances	Edges/Features
Graph	PPI [16]	3890	76584
	Twitch-EN [35]	7126	35324
	Facebook [35]	22,470	171,002
	AstroPh [27]	18772	198110
	CondMat [27]	23133	93497
Non-graph	Mushroom [37]	8124	22
	QSAR-oral[3]	8992	1024
	Occupancy [6]	20560	5

Similarity Measures. For graph data, we use both cosine similarity and Jaccard similarity to construct the similarity matrix [20]. For non-graph data, we use cosine similarity. For most of experiments, we follow the same strategy in [20] to keep the large entries in the similarity matrix that are above a threshold.

$$\text{threshold} = \text{mean}(\mathrm{S}) + \beta \text{std}(\mathrm{S}) \tag{28}$$

where mean(x) and std(x) calculate the mean and standard deviation of x, β is a predefined hyperparameter.

Metrics. For the sensitivity measure (Q1), we randomly perturb the similarity matrix and compare the output of Algorithm 1 ($\Delta\mathbf{Y}^*$) with that of re-training the learning model ($\mathbf{Y}^*_{\mathrm{S}+\Delta\mathrm{S}} - \mathbf{Y}^*_{\mathrm{S}}$). We use

$$scaleErr = |\log_{10}(\frac{\|\Delta\mathbf{Y}^*\|_\infty}{\|\mathbf{Y}^*_{\mathrm{S}+\Delta\mathrm{S}} - \mathbf{Y}^*_{\mathrm{S}}\|_\infty})| \tag{29}$$

$$errRate = \frac{\|(\mathbf{Y}^*_{\mathrm{S}+\Delta\mathrm{S}} - \mathbf{Y}^*_{\mathrm{S}}) - \Delta\mathbf{Y}^*\|_\infty}{\|\mathbf{Y}^*_{\mathrm{S}+\Delta\mathrm{S}} - \mathbf{Y}^*_{\mathrm{S}}\|_\infty} \tag{30}$$

to measure the accuracy. We also record the magnitude of outputs for a better comparison.

$$L_\infty(\text{Retrain}) = \|\mathbf{Y}^*_{\mathrm{S}+\Delta\mathrm{S}} - \mathbf{Y}^*_{\mathrm{S}}\|_\infty \quad L_\infty(\text{Soul-M}) = \|\Delta\mathbf{Y}^*\|_\infty \tag{31}$$

For robust algorithm (Q2), we use NDCG@50, NDCG@100 for PageRank and Accuracy for binary classification to measure utility.

$$NDCG@k = \frac{DCG@k}{IDCG@k} = \frac{\sum_{i=1}^{k} \frac{\mathbb{1}\{r[i] \in y\}}{\log_2(i+2)}}{\sum_{i=1}^{k} \frac{1}{\log_2(i+2)}} \tag{32}$$

where $\mathbb{1}\{\}$ is indicator function. We also use individual fairness bias in Eq. (1) to measure fairness. Sensitivity is measured by L_∞(Retrain) in Eq. (31). Given that we are the first to study sensitivity of individual fairness, We only consider the original in-processing algorithm in InFoRM [20] as baseline for the first question in Q2. To answer the second question of Q2, we use the ΔS learnt from Algorithm 2 to attack the similarity matrix and compare the utility and fairness after and before attacking. We also compare our attacking method with three baselines that attack the adjacency matrix (Gaussian noise (GN), PGD [28], and FGSM [15]) to show that attacking similarity measure is as effective as attacking graph structure for attacking model utility.

5.2 Main Results

For the sensitivity measure (Q1), results on PageRank and spectral clustering are shown in Tables 4 and 5, respectively. As we can see, our measurement can give good approximation of the change of learning outcomes. In most cases, scaleErr is smaller than 1, i.e., our approximation and the result via re-training are on the same scale. It is also worth noticing that Jaccard similarity is more stable than cosine similarity for PageRank, and cosine similarity is more stable than Jaccard similarity for spectral clustering. This result is consistent with our intuition: PageRank depends solely on the graph structure while spectral clustering often applies cosine similarity (or other vector-based distance measures) on $\mathbf{Y}^*$ (Eq. (17)) to cluster the nodes.

For the robust algorithm (Q2), results on binary classification and PageRank are shown in Tables 6 and 7, respectively. As we can see, the proposed Soul-A algorithm significantly lowers the sensitivity of learning models in all cases and effectively increases the models' performance when perturbed. Soul-A also improves the performance of unperturbed model in most cases. Meanwhile, Soul-A leads to a negligible increase of individual fairness bias. This is due to the trade-off between utility, fairness and robustness, which can be controlled by hyperparameters. Results for attacking PageRank are shown in Table 8. Our method reaches the best attacking performance in most cases, which demonstrates that the similarity measure can indeed be an effective source of attacking a learning model's utility. PGD and FGSM also lead to higher individual fairness bias, while our attacking method does not.

6 Related Works

In this section, we review the related works on two topics, including individual fairness and sensitivity analysis.

Individual fairness has received much attention in recent years. The definition of individual fairness was initially proposed in [10]. Although it considers fairness on an individual-based level, the intuition behind individual fairness, that similar individuals should be treated similarly, also benefits group fairness and counterfactual fairness. This is because individuals or groups with only diverse sensitive attributes will have similar outputs under this setting as well. Thereby, researchers have designed numerous methods to

Table 4: Effectiveness of sensitivity measure for PageRank. scaleErr and errRate are better if closer to 0.

Datasets	Jaccard Similarity				Cosine Similarity			
	L_∞(Retrain)	L_∞(Soul-M)	scaleErr	errRate	L_∞(Retrain)	L_∞(Soul-M)	scaleErr	errRate
PPI	2.8770×10^{-8}	5.3845×10^{-8}	0.2722	0.8716	1.0809×10^{-7}	1.7066×10^{-8}	0.8017	0.9971
Twitch-EN	3.0706×10^{-8}	5.5588×10^{-8}	0.2578	0.8103	33.0504	1.3230	1.3976	1.0007
Facebook	2.8552×10^{-9}	5.2685×10^{-9}	0.2661	0.7344	68.5519	8.1697	0.9238	0.9986
CondMat	4.3713×10^{-9}	7.7849×10^{-9}	0.2506	0.7809	6252542.2884	1250906.0486	0.6988	0.9957
AstroPh	2.3630×10^{-9}	4.2982×10^{-9}	0.2598	0.8427	6.0850×10^{-9}	3.3712×10^{-9}	0.2565	0.9978

Table 5: Effectiveness of sensitivity measure for spectral clustering. scaleErr and errRate are better if closer to 0.

Datasets	Jaccard Similarity				Cosine Similarity			
	L_∞(Retrain)	L_∞(Soul-M)	scaleErr	errRate	L_∞(Retrain)	L_∞(Soul-M)	scaleErr	errRate
PPI	2.3229×10^{-7}	2.1371×10^{-7}	0.0362	0.1386	1.2627×10^{-8}	1.2375×10^{-8}	0.0088	0.8821
Twitch-EN	1.0511×10^{-6}	1.0440×10^{-6}	0.0029	0.3355	1.3449×10^{-7}	2.4078×10^{-8}	0.7471	0.9999
Facebook	3.1688×10^{-8}	2.2686×10^{-8}	0.1451	0.2841	1.7938×10^{-9}	1.7292×10^{-9}	0.0159	0.3345
CondMat	1.0548×10^{-7}	9.9071×10^{-8}	0.0272	0.9090	1.2200×10^{-6}	2.0876×10^{-7}	0.7667	1.0000
AstroPh	1.0531×10^{-7}	3.0376×10^{-8}	0.5399	0.9981	2.8077×10^{-8}	2.6811×10^{-8}	0.0200	0.0451

Table 6: Effectiveness of robustification for binary classification. Lower is better in gray columns, larger is better in the rest. 'w.p.' is abbreviation for 'with perturbation'.

Datasets	Method	Cosine Similarity				
		Sensitivity	Accuracy	Accuracy w.p.	Bias	Bias w.p.
Mushroom	InFoRM	0.0853	1.0	1.0	4.4124×10^{6}	4.4300×10^{6}
	Ours	0.0813	1.0	1.0	4.4124×10^{6}	4.4300×10^{6}
QSAR-oral	InFoRM	0.1467	0.9311	0.9271	9.9139×10^{5}	1.0353×10^{6}
	Ours	0.1427	0.9316	0.9273	1.0007×10^{6}	1.0352×10^{6}
occupancy	InFoRM	0.2532	0.9894	0.9714	1.2371×10^{7}	1.2443×10^{7}
	Ours	0.2170	0.9907	0.9753	1.3042×10^{7}	1.4152×10^{7}

Table 7: Effectiveness of robustification for PageRank. Lower is better in gray columns, larger is better in the rest. 'w.p.' is abbreviation for 'with perturbation'.

Datasets	Method	Jaccard Similarity						
		Sensitivity	NDCG@50	NDCG@50 w.p.	NDCG@100	NDCG@100 w.p.	Bias	Bias w.p.
PPI	InFoRM	7.8024×10^{-4}	0.8048	0.6964	0.8657	0.7635	3.5931×10^{-6}	3.7444×10^{-7}
	Ours	2.6131×10^{-6}	0.8048	0.8048	0.8668	0.8657	3.6696×10^{-6}	3.6326×10^{-6}
Twitch-EN	InFoRM	6.8099×10^{-4}	0.8891	0.8270	0.9301	0.8020	9.8781×10^{-7}	1.6375×10^{-7}
	Ours	6.1682×10^{-7}	0.8891	0.8891	0.9334	0.9301	1.0015×10^{-6}	1.0053×10^{-6}
Facebook	InFoRM	1.4002×10^{-4}	0.8580	0.6776	0.7625	0.7601	6.0219×10^{-7}	5.7810×10^{-8}
	Ours	8.7429×10^{-7}	0.8492	0.8580	0.7638	0.7390	6.1930×10^{-7}	6.1909×10^{-7}
CondMat	InFoRM	1.3973×10^{-4}	0.8667	0.7143	0.8030	0.7664	4.4756×10^{-7}	4.1871×10^{-8}
	Ours	1.9225×10^{-7}	0.8667	0.8667	0.8206	0.8206	4.5677×10^{-7}	4.5545×10^{-7}
AstroPh	InFoRM	9.6574×10^{-5}	0.8849	0.6600	0.9036	0.5963	7.5825×10^{-7}	8.0497×10^{-8}
	Ours	5.6437×10^{-7}	0.9135	0.9135	0.9202	0.9149	7.7854×10^{-7}	7.7868×10^{-7}

Datasets	Method	Cosine Similarity						
		Sensitivity	NDCG@50	NDCG@50 w.p.	NDCG@100	NDCG@100 w.p.	Bias	Bias w.p.
PPI	InFoRM	6.4236×10^{-4}	0.8844	0.8587	0.8453	0.7365	6.5310×10^{-6}	1.1296×10^{-6}
	Ours	7.5255×10^{-7}	0.8844	0.8844	0.8456	0.8456	6.5570×10^{-6}	6.5537×10^{-6}
Twitch-EN	InFoRM	4.5977×10^{-4}	0.8916	0.459	0.8714	0.8645	3.2911×10^{-6}	6.0439×10^{-7}
	Ours	4.1129×10^{-7}	0.8916	0.8916	0.8714	0.8714	3.3034×10^{-6}	3.3033×10^{-6}
Facebook	InFoRM	1.1551×10^{-4}	0.8181	0.8818	0.7982	0.7639	9.8483×10^{-7}	1.2171×10^{-7}
	Ours	5.4312×10^{-7}	0.8181	0.8181	0.8497	0.7959	9.9666×10^{-7}	9.9645×10^{-7}
CondMat	InFoRM	1.0228×10^{-4}	0.6833	0.6841	0.7281	0.8160	8.5178×10^{-7}	1.0779×10^{-7}
	Ours	1.7541×10^{-7}	0.6833	0.6833	0.7283	0.7281	8.5962×10^{-7}	8.5873×10^{-7}
AstroPh	InFoRM	6.5637×10^{-5}	0.8281	0.6732	0.7709	0.6221	1.1006×10^{-6}	1.4903×10^{-7}
	Ours	1.0701×10^{-7}	0.8695	0.8008	0.7685	0.7711	1.1164×10^{-6}	1.1117×10^{-6}

apply individual fairness in various problem settings. Sepideh et al. [29] give an individually fair k-clustering algorithm that expects each point's fair radius to be on the same scale. Asia et al. [4] work on solving individual fairness problem in recommender systems or other settings that require ranking. They define individual fairness on rankings and add attention to the framework by utilizing cumulative relevance. Giuseppe et al. propose fAux [7] which defines 'similar' as differing only on protected variables, and it ensures individual fairness by testing the derivative of the predictions of the model with an auxiliary model that predicts protected variables. InFoRM [20] investigates individual fairness on various graph mining tasks, with some follow-up works on individual fairness in graph neural networks (GNNs). Dong et al. [9] study ranking based individual fairness and propose a plug-and-play framework, REDRESS, to balance GNNs' utility and fairness. Xu et al. [43] implement an auxiliary link prediction task on a fairness graph which is generated from the similarity matrix, and then concatenate the outputs, called fairness hint, with the embeddings generated by original GNNs for downstream tasks. Song et al. propose GUIDE [38] that aims at

Table 8: Effectiveness of attack for PageRank. Higher denotes better attacking result in gray column, lower is better in the rest.

Datasets	Methods	Jaccard Similarity			Cosine Similarity		
		NDCG@50	NDCG@100	Bias	NDCG@50	NDCG@100	Bias
PPI	Vanilla	0.8048	0.8657	3.5931×10^{-6}	0.8844	0.8453	6.5310×10^{-6}
	GN	0.7803	0.7965	7.0030×10^{-6}	0.7274	0.8309	9.9342×10^{-6}
	PGD	0.7324	0.6563	3.9540×10^{14}	0.7573	0.6918	4.3089×10^{10}
	FGSM	0.7402	0.6819	2.2798×10^{14}	0.5266	0.3775	3.3067×10^{-6}
	Ours	0.6855	0.7646	8.4277×10^{-7}	0.6663	0.7802	3.9952×10^{-6}
Twitch-EN	Vanilla	0.8891	0.9301	9.8781×10^{-7}	0.8916	0.8714	3.2911×10^{-6}
	GN	0.8424	0.8622	1.2211×10^{-6}	0.8089	0.8039	3.7858×10^{-6}
	PGD	0.7125	0.7799	3.3603×10^{2}	0.7542	0.8494	6.1631×10^{-1}
	FGSM	0.7125	0.7778	3.2610×10^{2}	0.7815	0.7814	2.0659×10^{-5}
	Ours	0.6655	0.5495	7.1624×10^{-8}	0.8833	0.8260	1.1824×10^{-6}
Facebook	Vanilla	0.8580	0.7625	6.0219×10^{-7}	0.8181	0.7982	9.8483×10^{-7}
	GN	0.4829	0.4936	9.7690×10^{-7}	0.4624	0.5536	1.4124×10^{-6}
	PGD	0.2438	0.2162	1.2588×10^{23}	0.2198	0.2097	4.0474×10^{20}
	FGSM	0.2491	0.2176	7.9818×10^{22}	0.0	0.2303	2.0527×10^{8}
	Ours	0.6477	0.5607	7.9161×10^{-8}	0.4644	0.4797	1.1498×10^{-7}
CondMat	Vanilla	0.8667	0.8030	4.4756×10^{-7}	0.6833	0.7281	8.5178×10^{-7}
	GN	0.6957	0.7514	4.6734×10^{-7}	0.6784	0.7309	8.8630×10^{-7}
	PGD	0.2595	0.3136	7.0861×10^{-3}	0.0	0.0566	9.7560×10^{-7}
	FGSM	0.2626	0.3399	3.7412×10^{-3}	0.0198	0.0408	4.5741×10^{-7}
	Ours	0.7572	0.7116	4.7817×10^{-8}	0.6344	0.6438	1.3941×10^{-7}
AstroPh	Vanilla	0.8849	0.9036	7.5825×10^{-7}	0.8281	0.7709	1.1006×10^{-6}
	GN	0.7804	0.6473	1.3406×10^{-6}	0.7598	0.6456	1.6992×10^{-6}
	PGD	0.5105	0.5037	7.1157×10^{17}	0.4181	0.3485	6.9490×10^{14}
	FGSM	0.5102	0.5144	2.3727×10^{17}	0.0	0.0072	1.5438×10^{7}
	Ours	0.3664	0.3661	1.9967×10^{-7}	0.4279	0.5601	6.0060×10^{-8}

equalizing individual fairness among groups while achieving individual fairness. They define Group Equality Informed Individual Fairness and use the attention mechanism in graph attention networks [39]. Yurochkin et al. [44] consider individual fairness from the robustness perspective, interpreting it as performance invariance under sensitive perturbations to inputs. There also exist works focusing on attacking individual fairness, e.g., FATE [22] attacks individual fairness on graphs through generating poisoned adjacency and feature matrix using meta-gradient. To our best knowledge, we are the first to study study the sensitivity of the individual fairness and attacking model utility through similarity matrix.

Sensitivity analysis (SA) is the study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input [36]. Understanding sensitivity of a model can help evaluate robustness and uncertainty of the model, explain how the input impacts the output, and reveal critical latent relationships. This field of study first caught researchers' eyes back in late twentieth century and numerous methods to conduct sensitivity analysis in different scenarios have been proposed ever since. Webster et al. [34] reference methods from visual psychophysics that uses manual stimulation and assessment of the response to understand how a failure of a model is caused by an individual and promote face recognition's explainability. Kala [19] analyzes failure of a bridge with respect to several input variables. Zhang et al. [46] apply Sobol's method to analyze how parameters contribute to the outputs of a covid-19 pandemic mathematical model. N2N [21] defines the network derivative mining problem that analyzes how edges of the input graph influence the mining results. There are previous works that use sensitivity analysis to understand different types of fairness. Ghosh et al.[14] define fairness influence function (FIF) to quantify the influence of input features on a classifier's bias. They also design the FairXplainer that calculates FIF as difference of conditional variances. Joshi et al.[18] focus on sensitivity analysis in face recognition. They extend VPSA [34] to analyze how fair the model's performance in different subgroups is when perturbed.

7 Conclusion

In this paper, we formally introduce the sensitivity analysis problem of individual fairness, which quantifies how individually fair learning outcome is affected by the imperfection of the similarity matrix. We first quantify the gradient of individually fair learning outcome with respect to the similarity matrix with help of influence function, which allows us to approximate the change of individually fair learning outcome when the similarity matrix is perturbed. We further propose a robust individually fair algorithm via nested bi-level optimization. Our sensitivity measure and robust algorithm are applicable to a broad range of learning models as long as the objective function is twice differentiable. We instantiate the proposed sensitivity measure and robust algorithm with three popular learning models, including ranking, clustering and classification. Extensive experiments are conducted to demonstrate the efficacy of our methods. This work contributes to the broader problem that how robust is existing individually fair algorithms with respect to similarities, i.e., are existing works stable with inaccurate similarities. Furthermore, our work paves the way for finding a more reliable similarity measure by robustifying individual fairness algorithm. Moving forward, future work could focus on certified robustness in individual fairness.

Acknowledgments

This work is supported by NSF (1947135), NIFA (2020-67021-32799), DHS (17STQAC00001-07-00), and IBM-Illinois Discovery Accelerator Institute. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[3] Davide Ballabio, Francesca Grisoni, Viviana Consonni, and Roberto Todeschini. 2019. Integrated QSAR models to predict acute oral systemic toxicity. *Molecular informatics* 38, 8-9 (2019), 1800124.

[4] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 405–414. https://doi.org/10.1145/3209978.3210063

[5] Dan G Cacuci, Mihaela Ionescu-Bujor, and Ionel Michael Navon. 2005. *Sensitivity and uncertainty analysis, volume II: applications to large-scale systems*. Vol. 2. CRC press.

[6] Luis M Candanedo and Véronique Feldheim. 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings* 112 (2016), 28–39.

[7] Giuseppe Castiglione, Ga Wu, Christopher Srinivasa, and Simon Prince. 2022. fAux: Testing Individual Fairness via Gradient Alignment. *arXiv preprint arXiv:2210.06288* (2022).

[8] Enyan Dai and Suhang Wang. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 680–688.

[9] Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. 2021. Individual fairness for graph neural networks: A ranking based approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 300–310.

[10] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (Cambridge, Massachusetts) *(ITCS'12)*. Association for Computing Machinery, New York, NY, USA, 214–226. https://doi.org/10.1145/2090236.2090255

[11] Cynthia Dwork, Nicole Immorlica, Adam Tauman Kalai, and Max Leiserson. 2018. Decoupled Classifiers for Group-Fair and Efficient Machine Learning. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81)*, Sorelle A. Friedler and Christo Wilson (Eds.). PMLR, 119–133. https://proceedings.mlr.press/v81/dwork18a.html

[12] Will Fleisher. 2021. What's Fair about Individual Fairness?. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society* (Virtual Event, USA) *(AIES '21)*. Association for Computing Machinery, New York, NY, USA, 480–490. https://doi.org/10.1145/3461702.3462621

[13] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H. Chi, and Alex Beutel. 2019. Counterfactual Fairness in Text Classification through Robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (Honolulu, HI, USA) *(AIES '19)*. Association for Computing Machinery, New York, NY, USA, 219–226. https://doi.org/10.1145/3306618.3317950

[14] Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S Meel. 2022. How Biased is Your Feature?: Computing Fairness Influence Functions with Global Sensitivity Analysis. *arXiv preprint arXiv:2206.00667* (2022).

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[17] Christina Ilvento. 2019. Metric learning for individual fairness. *arXiv preprint arXiv:1906.00250* (2019).

[18] Aparna R Joshi, Xavier Suau Cuadros, Nivedha Sivakumar, Luca Zappella, and Nicholas Apostoloff. 2022. Fair SA: Sensitivity Analysis for Fairness in Face Recognition. In *Algorithmic Fairness through the Lens of Causality and Robustness workshop*. PMLR, 40–58.

[19] Zdeněk Kala. 2019. Global sensitivity analysis of reliability of structural bridge system. *Engineering Structures* 194 (2019), 36–45. https://doi.org/10.1016/j.engstruct.2019.05.045

[20] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. InFoRM: Individual Fairness on Graph Mining *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 379–389. https://doi.org/10.1145/3394486.3403080

[21] Jian Kang and Hanghang Tong. 2019. N2n: Network derivative mining. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 861–870.

[22] Jian Kang, Yinglong Xia, Ross Maciejewski, Jiebo Luo, and Hanghang Tong. 2023. Deceptive Fairness Attacks on Graphs via Meta Learning. *arXiv preprint arXiv:2310.15653* (2023).

[23] Jian Kang, Qinghai Zhou, and Hanghang Tong. 2022. JuryGCN: quantifying jackknife uncertainty on graph convolutional networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 742–752.

[24] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.

[25] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual Fairness. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf

[26] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. 2019. iFair: Learning Individually Fair Data Representations for Algorithmic Decision Making. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1334–1345. https://doi.org/10.1109/ICDE.2019.00121

[27] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[29] Sepideh Mahabadi and Ali Vakilian. 2020. Individual fairness for k-clustering. In *International Conference on Machine Learning*. PMLR, 6586–6596.

[30] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (jul 2021), 35 pages. https://doi.org/10.1145/3457607

[31] Debarghya Mukherjee, Mikhail Yurochkin, Moulinath Banerjee, and Yuekai Sun. 2020. Two simple ways to learn individual fairness metrics from data. In *International Conference on Machine Learning*. PMLR, 7097–7107.

[32] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001).

[33] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[34] Brandon RichardWebster, So Yon Kwon, Christopher Clarizio, Samuel E Anthony, and Walter J Scheirer. 2018. Visual psychophysics for making face recognition algorithms more explainable. In *Proceedings of the European conference on computer vision (ECCV)*. 252–270.

[35] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. arXiv:1909.13021 [cs.LG]

[36] Andrea Saltelli. 2002. Sensitivity Analysis for Importance Assessment. *Risk Analysis* 22, 3 (2002), 579–590. https://doi.org/10.1111/0272-4332.00040 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00040

[37] Jeffrey Curtis Schlimmer. 1987. *Concept acquisition through representational adjustment.* University of California, Irvine.

[38] Weihao Song, Yushun Dong, Ninghao Liu, and Jundong Li. 2022. GUIDE: Group Equality Informed Individual Fairness in Graph Neural Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1625–1634. https://doi.org/10.1145/3534678.3539346

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[40] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2023–2040.

[41] Eric W Weisstein. 2003. Gershgorin circle theorem. *https://mathworld. wolfram.com/* (2003).

[42] Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. 2021. Learning Fair Representations for Recommendation: A Graph-Based Perspective. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2198–2208. https://doi.org/10.1145/3442381.3450015

[43] Paiheng Xu, Yuhang Zhou, Bang An, Wei Ai, and Furong Huang. 2022. GFairHint: Improving Individual Fairness for Graph Neural Networks via Fairness Hint. In *Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022*. https://openreview.net/forum?id=DCQmL-gXGOG

[44] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. 2020. Training individually fair ML models with sensitive subspace robustness. (2020).

[45] Richard S. Zemel, Ledell Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. 2013. Learning Fair Representations. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:490669

[46] Zizhen Zhang, Raheem Gul, and Anwar Zeb. 2021. Global sensitivity analysis of COVID-19 mathematical model. *Alexandria Engineering Journal* 60, 1 (2021), 565–572. https://doi.org/10.1016/j.aej.2020.09.035

[47] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.