Poisoning Attacks on Algorithmic Fairness

David Solans[1][0000−0001−6979−9330], Battista Biggio[2][0000−0001−7752−509X], and Carlos Castillo[3][0000−0003−4544−0416]

[1] Universitat Pomepu Fabra, Barcelona, Spain
{firstname, lastname}@upf.edu
[2] Università degli Studi di Cagliari, Cagliari, Italy
{firstname,lastname}@unica.it
[3] Universitat Pomepu Fabra, Barcelona, Spain
chato@acm.org

Abstract. Research in adversarial machine learning has shown how the performance of machine learning models can be seriously compromised by injecting even a small fraction of poisoning points into the training data. While the effects on model accuracy of such poisoning attacks have been widely studied, their potential effects on other model performance metrics remain to be evaluated. In this work, we introduce an optimization framework for poisoning attacks against algorithmic fairness, and develop a gradient-based poisoning attack aimed at introducing classification disparities among different groups in the data. We empirically show that our attack is effective not only in the white-box setting, in which the attacker has full access to the target model, but also in a more challenging black-box scenario in which the attacks are optimized against a substitute model and then transferred to the target model. We believe that our findings pave the way towards the definition of an entirely novel set of adversarial attacks targeting algorithmic fairness in different scenarios, and that investigating such vulnerabilities will help design more robust algorithms and countermeasures in the future.

Keywords: algorithmic discrimination · algorithmic fairness · poisoning attacks · adversarial machine learning · machine learning security

1 Introduction

Algoritmic Fairness is an emerging concern in computing science that started within the data mining community but has extended into other fields including machine learning, information retrieval, and theory of algorithms [11]. It deals with the design of algorithms and decision support systems that are non-discriminatory, i.e., that do not introduce an unjustified disadvantage for members of a group, and particularly that do not further place at a disadvantage members of an already disadvantaged social group. In machine learning, the problem that has been most studied to date is supervised classification, in which algorithmic fairness methods have been mostly proposed to fulfill criteria related to parity (equality) [24]. Most of the methods proposed to date assume benevolence from the part of the data scientist or developer creating the classification model: she is envisioned as an actor trying to eliminate or reduce potential discrimination in her model.

The problem arises when dealing with malicious actors that can tamper with the model development, for instance by tampering with training data. Traditionally, *poisoning attacks* have been studied in *Adversarial Machine Learning*. These attacks are usually crafted with the purpose of increasing the misclassification rate in a machine learning model, either for certain samples or in an indiscriminate basis, and have been widely demonstrated in adversarial settings (see, e.g., [4]).

In this work, we show that an attacker may be able to introduce algorithmic discrimination by developing a novel poisoning attack. The purpose of this attacker is to create or increase a disadvantage against a specific group of individuals or samples. For that, we explore how analogous techniques can be used to compromise a machine learning model, not to drive its accuracy down, but with the purpose of adding algorithmic discrimination, or exaggerating it if it already exists. In other words, the purpose of the attacker will be to create or increase a disadvantage against a specific group of individuals or samples.

Motivation. The main goal of this paper is to show the potential harm that an attacker can cause in a machine learning system if the attacker can manipulate its training data. For instance, the developer of a criminal recidivism prediction tool [1] could sample training data in a discriminatory manner to bias the tool against a certain group of people. Similar harms can occur when training data is collected from public sources, such as online surveys that cannot be fully trusted. A minority of ill-intentioned users could *poison* this data to introduce defects in the machine learning system created from it. In addition to these examples, there is the unintentional setting, where inequities are introduced in the machine learning model as an undesired effect of the data collection or data labeling. For instance, human annotators could systematically make mistakes when assigning labels to images of people of a certain skin color [6].

The methods we describe on this paper could be used to model the potential harm to a machine learning system in the worst-case scenario, demonstrating the undesired effects that a very limited amount of wrongly labeled samples can cause, even if created in an unwanted manner.

Contributions. This work first introduces a novel optimization framework to craft poisoning samples that against algorithmic fairness. After this, we perform experiments in two scenarios: a "black-box" attack in which the attacker only has access to a set of data sampled from the same distribution as the original training data, but not the model nor the original training set, and a "white-box" scenario in which the attacker has full access to both. The effects of these attacks are measured using impact quantification metrics. The experiments show that by carefully perturbing a limited amount of training examples, an skilled attacker has the possibility of introducing different types of inequities for certain groups of individuals. This, can be done without large effects on the overall accuracy of the system, which makes these attacks harder to detect.

Paper structure. The rest of this paper is organized as follows. Section 2, describes the proposed methodology to craft poisoning attacks for algorithmic fairness. Section 3 demonstrates empirically the feasibility of the new types of attacks on both synthetic and real-world data, under different scenarios depending on the attacker knowledge about the system. Section 4 provides further references to related work. Section 5 presents our conclusions.

2 Poisoning Fairness

In this section we present a novel gradient-based poisoning attack, crafted with the purpose of compromising algorithmic fairness, ideally without significantly degrading accuracy.

Notation. Feature and label spaces are denoted in the following with $\mathcal{X} \subseteq \mathbb{R}^{\mathsf{d}}$ and $\mathcal{Y} \in \{-1, 1\}$, respectively, with d being the dimensionality of the feature space. We assume that the attacker is able to collect some training and validation data sets that will be used to craft the attack. We denote them as $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$. Note that these sets include samples along with their labels. $L(\mathcal{D}_{\text{val}}, \theta)$ is used to denote the validation loss incurred by the classifier $f_\theta : \mathcal{X} \to \mathcal{Y}$, parametrized by θ, on the validation set $\mathcal{D}_{\text{val}}$. $\mathcal{L}(\mathcal{D}_{\text{tr}}, \theta)$ is used to represent the regularized loss optimized by the classifier during training.

2.1 Attack Formulation

Using the aforementioned notation, we can formulate the optimal poisoning strategy in terms of the following bilevel optimization:

$$\max_{\mathbf{x}_c} \; \mathcal{A}(\mathbf{x}_c, y_c) = L(\mathcal{D}_{\text{val}}, \theta^\star) \, , \tag{1}$$

$$\text{s.t.} \; \theta^\star \in \arg\min_{\theta} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup (\mathbf{x}_c, y_c), \theta) \, , \tag{2}$$

$$\mathbf{x}_{\text{lb}} \preceq \mathbf{x}_c \preceq \mathbf{x}_{\text{ub}} \, . \tag{3}$$

The goal of this attack is to maximize a loss function on a set of untainted (validation) samples, by optimizing the poisoning sample $\mathbf{x}_c$, as stated in the outer optimization problem (Eq. 1). To this end, the poisoning sample is labeled as y_c and added to the training set $\mathcal{D}_{\text{tr}}$ used to learn the classifier in the inner optimization problem (Eq. 2). As one may note, the classifier $\theta^\star$ is learned on the poisoned training data, and then used to compute the outer validation loss. This highlights that there is an implicit dependency of the outer loss on the poisoning point $\mathbf{x}_c$ via the optimal parameters $\theta^\star$ of the trained classifier. In other words, we can express the optimal parameters $\theta^\star$ as a function of $\mathbf{x}_c$, i.e., $\theta^\star(\mathbf{x}_c)$. This relationship tells us how the classifier parameters change when the poisoning point $\mathbf{x}_c$ is perturbed. Characterizing and being able to manipulate this behavior is the key idea behind poisoning attacks.

Within this formulation, additional constraints on the feature representation of the poisoning sample can also be enforced, to make the attack samples stealthier or more difficult to detect. In this work we only consider a box constraint that requires the feature values of $\mathbf{x}_c$ to lie within some lower and upper bounds (in Eq. 3, the operator $\preceq$ enforces the constraint for each value of the feature vectors involved). This constraint allows us to craft poisoning samples that lie within the feature values observed in the training set. Additional constraints can be additionally considered, e.g., constraints imposing a maximum distance from an initial location or from samples of the same class, we leave their investigation to future work. Our goal here is to evaluate the extent to which a poisoning attack which is only barely constrained can compromise algorithmic fairness.

The bilevel optimization considered here optimizes one poisoning point at a time. To optimize multiple points, one may inject a set of properly-initialized attack points into the training set, and then iteratively optimize them one at a time. Proceeding on a greedy fashion, one can add and optimize one point at a time, sequentially. This strategy is typically faster but suboptimal (as each point is only optimized once, and may become suboptimal after injection and optimization of the subsequent points).

Attacking Algorithmic Fairness. We now define an objective function $\mathcal{A}(\mathbf{x}_c, y_c)$ in terms of a validation loss $L(\mathcal{D}_{\text{val}}, \theta)$ that will allow us to compromise algorithmic fairness without significantly affecting classification accuracy. To this end, we consider the *disparate impact* criterion [3]. This criterion assumes data items, typically representing individuals, can be divided into unprivileged (e.g., people with a disability) and privileged (e.g., people without a disability), and that there is a positive outcome (e.g., being selected for a scholarship). Disparate impact is observed when the fraction of unprivileged people obtaining the positive outcome is much lower the fraction of privileged people obtaining the positive outcome. Formally, to avoid disparate impact:

$$D = \frac{P(\hat{Y} = 1 | G = u)}{P(\hat{Y} = 1 | G = p} \geq 1 - \epsilon \,, \tag{4}$$

where $\hat{Y}$ is the predicted label, and $G = \{u, p\}$ a *protected attribute* denoting the group of unprivileged (u) and privileged (p) samples within a set $\mathcal{D}$. Disparate impact thus measures the ratio between the fractions of unprivileged and privileged samples that are assigned to the positive class. Typically, one sets $\epsilon \approx 0.2$ which suggests $D \geq 0.8$ for a fair classifier, as stated by the four-fifths rule of maximum acceptable disparate impact proposed by the US Equal Employment Opportunity Commission (EEOC) [10,25]. Thus, in general, we should have D values closer to one to improve fairness.

For our poisoning attack to work, we aim to minimize such a ratio, i.e., decreasing the fraction of unprivileged samples for which $\hat{y} = 1$, while increasing the fraction of privileged users which are assigned $\hat{y} = 1$. For numerical convenience, we choose to maximize the difference (instead of the ratio) between the mean loss computed on the unprivileged and the privileged samples:

$$L(\mathcal{D}_{\text{val}}, \theta) = \underbrace{\sum_{k=1}^{p} \ell(\mathbf{x}_k, y_k, \theta)}_{\text{unprivileged}} + \lambda \underbrace{\sum_{j=1}^{m} \ell(\mathbf{x}_j, y_j, \theta)}_{\text{privileged}} \,. \tag{5}$$

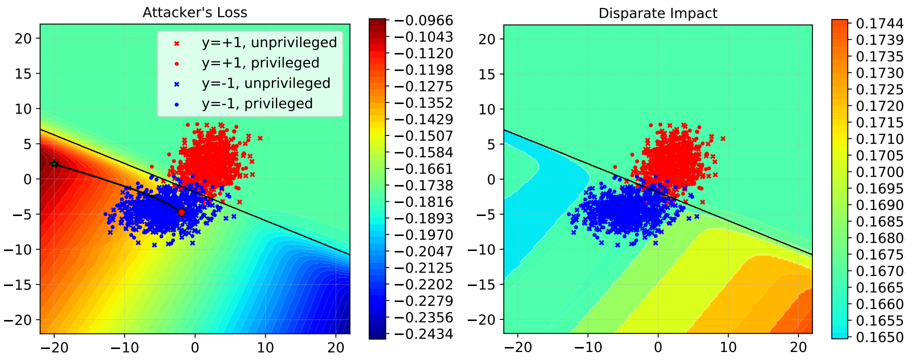

Fig. 1. Attacker's loss $\mathcal{A}(\mathbf{x}_c, y_c)$ (*left*) and disparate impact (*right*) as a function of the attack point $\mathbf{x}_c$ with $y_c = 1$, on a bi-dimensional classification task. Note how the attacker's loss provides a smoother approximation of the disparate impact, and how our gradient-based attack successfully optimizes the former, which amounts to minimizing disparate impact, compromising algorithmic fairness.

Note that the parameter λ here is set to p/m to balance the class priors (rather than dividing the first term by p and the second by m).

To minimize D, we would like to have unprivileged samples classified as negative (lower numerator) and privileged classified as positive (higher denominator). As we aim to maximize $L(\mathcal{D}_{\text{val}}, \theta)$, we can label the unprivileged samples as positive ($y_k = 1$), and the privileged samples as negative ($y_j = -1$). Maximizing this loss will enforce the attack to increase the number of unprivileged samples classified as negative and of privileged samples classified as positive.

In Fig. 1, we report a comparison of the attacker's loss $\mathcal{A}(\mathbf{x}_c, y_c) = L(\mathcal{D}_{\text{val}}, \theta^\star)$ as given by Eq. (5) and the disparate impact D, as a function of the attack point $\mathbf{x}c$ (with $y_c = 1$) in a bi-dimensional toy example. Each point in the plot represents the value of the function (either $\mathcal{A}$ or D computed on an untainted validation set) when the point $\mathbf{x}_c$ corresponding to that location is added to the training set. These plots show that our loss function provides a nice smoother approximation of the disparate impact, and that maximizing it correctly amounts to minimizing disparate impact, thus compromising algorithmic fairness.

2.2 Gradient-Based Attack Algorithm

Having defining our (outer) objective, we are now in the position to discuss how to solve the given bilevel optimization problem. Since our objective is differentiable, we can make use of existing gradient-based strategies to tackle this problem. In particular, we will use a simple gradient ascent strategy with projection (to enforce the box constraint of Eq. 3). The complete algorithm is given as Algorithm 1. In Fig. 1 we also report an example of how this algorithm is able to find a poisoning point that maximizes the attacker's loss.

Algorithm 1 Gradient-based poisoning attack

```
Require: $\mathbf{x}_c, y_c$: the initial location of the poisoning sample and its label; $\eta$: the gradient step size; $t > 0$: a small number.
Ensure: $\mathbf{x}'_c$: the optimized poisoning sample.
1: Initialize the attack sample: $\mathbf{x}'_c \leftarrow \mathbf{x}_c$
2: repeat
3:     Store attack from previous iteration: $\mathbf{x}_c \leftarrow \mathbf{x}'_c$
4:     Update step: $\mathbf{x}'_c \leftarrow \Pi\left(\mathbf{x}_c + \eta \nabla_{\mathbf{x}_c} \mathcal{A}\right)$, where $\Pi$ ensures projection onto the feasible domain (i.e., the box constraint in Eq. 3).
5: until $|\mathcal{A}(\mathbf{x}'_c, y_c) - \mathcal{A}(\mathbf{x}_c, y_c)| \leq t$
6: return $\mathbf{x}'_c$
```

Attack Initialization. An important remark to be made here is that *initialization* of the poisoning samples plays a key role. In particular, if we initialize the attack point as a point which is correctly classified by the algorithm, the attack will not even probably start at all. This is clear if one looks at Fig. 1, where we consider an attack point labeled as positive (red). If we had initialized the point in the top-right area of the figure, where positive (red) points are correctly classified, the point would have not even moved from its initial location, as the gradient in that region is essentially zero (the value of the objective is constant). Hence, for a poisoning attack to be optimized properly, a recommended strategy is to initialize points by sampling from the available set at random, but then flipping their label. This reduces the risk of starting from a flat region with null gradients [5,22].

Gradient Computation. Despite the simplicity of the given projected gradient-ascent algorithm, the computation of the poisoning gradient $\nabla_{\mathbf{x}_c}\mathcal{A}$ is more complicated. In particular, we do not only need the outer objective to be sufficiently smooth w.r.t. the classification function, but also the solution $\theta^\star$ of the inner optimization to vary smoothly with respect to $\mathbf{x}_c$ [5,17,8,4]. In general, we need $\mathcal{A}$ to be sufficiently smooth w.r.t. $\mathbf{x}_c$.

Under this assumption, the gradient can be obtained as follows. First, we derive the objective function w.r.t. $\mathbf{x}_c$ using the chain rule [5,22,17,4,15]:

$$\nabla_{\mathbf{x}_c}\mathcal{A} = \nabla_{\mathbf{x}_c} L + \frac{\partial \theta^\star}{\partial \mathbf{x}_c}^\top \nabla_\theta L\,, \tag{6}$$

where the term $\frac{\partial \theta^\star}{\partial \mathbf{x}_c}$ captures the implicit dependency of the parameters θ on the poisoning point $\mathbf{x}$, and $\nabla_{\mathbf{x}_c} L$ is the explicit derivative of the outer validation loss w.r.t. $\mathbf{x}_c$. Typically, this is zero if $\mathbf{x}_c$ is not directly involved in the computation of the classification function f, e.g., if a linear classifier is used (for which $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$). In the case of kernelized SVMs, instead, there is also an explicit dependency of L on $\mathbf{x}_c$, since it appears in the computation of the classification function f when it joins the set of its support vectors (see, e.g., [5,8]).

Under regularity of $\theta^\star(\mathbf{x}_c)$, the derivative $\frac{\partial \theta^\star}{\partial \mathbf{x}_c}$ can be computed by replacing the inner optimization problem in Eq. (2) with its equilibrium (Karush-Kuhn-Tucker, KKT) conditions, i.e., with the implicit equation $\nabla_\theta \mathcal{L}(\mathcal{D}_{\rm tr} \cup (\mathbf{x}_c, y_c), \theta) \in$

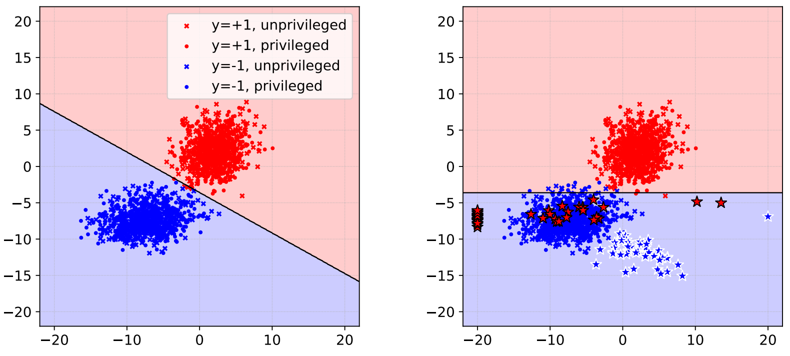

Fig. 2. Gradient-based poisoning attack against a logistic classifier, on a bi-dimensional classification task. The classification function and the corresponding decision regions are reported before (*left*) and after (*right*) injection of the poisoning samples (red and blue stars in the right plot).

$\mathbf{0}$ [15,17]. By deriving this expression w.r.t. $\mathbf{x}_c$, we get a linear system of equations, expressed in matrix form as $\nabla_{\mathbf{x}_c} \nabla_\theta \mathcal{L} + \frac{\partial \theta^\star}{\partial \mathbf{x}}^\top \nabla^2_{\mathbf{w}} \mathcal{L} \in \mathbf{0}$. We can now compute $\frac{\partial \theta^\star}{\partial \mathbf{x}_c}$ from these equations, and substitute the result in Eq. (6), obtaining the required gradient:

$$\nabla_{\mathbf{x}_c} \mathcal{A} = \nabla_{\mathbf{x_c}} L - (\nabla_{\mathbf{x}_c} \nabla_\theta \mathcal{L})(\nabla^2_\theta \mathcal{L})^{-1} \nabla_\theta L \, . \tag{7}$$

These gradients can be computed for various classifiers (see, e.g., [8]). In our case, we simply need to compute the term $\nabla_\theta L$, to account for the specific validation loss that we use to compromise algorithmic fairness (Eq. 5).

Finally, in Fig. 2, we show how our poisoning attack modifies the decision function of a linear classifier to worsen algorithmic fairness on a simple bi-dimensional example. As one may appreciate, the boundary is slightly tilted, causing more unprivileged samples to be classified as negative, and more privileged samples to be classified as positive.

2.3 White-Box and Black-Box Poisoning Attacks

The attack derivation and implementation discussed throughout this section implicitly assumes that the attacker has full knowledge of the attacked system, including the training data, the feature representation, and the learning and classification algorithms. This sort of *white-box* access to the targeted system is indeed required to compute the poisoning gradients correctly and run the poisoning attack [4]. It is however possible to also craft *black-box* attacks against different classifiers by using essentially the same algorithm. To this end, one needs to craft the attacks against a *surrogate model*, and then check if these attack samples *transfer* successfully to the actual target model. Interestingly, in many cases these black-box transfer attacks have been shown to work effectively, provided that the surrogate model is sufficiently similar to the target ones [19,8]. The underlying assumption here is that it is possible to train the surrogate model on samples drawn from the same distribution as those used by the target model, or that sufficient queries can be sent to the target model to reconstruct its behavior.

In our experiments we consider both white-box attacks and black-box transfer attacks to also evaluate the threat of poisoning fairness against weaker attackers that only possess limited knowledge of the target model. For black-box attacks, in particular, we assume that the attacker trains the substitute models on a training set sampled from the same distribution as that of the target models, but no queries are sent to the target classifiers while optimizing the attack.

3 Experiments

This section describes the obtained results for two different datasets, one synthetic set composed of 2000 samples, each of them having three features, one of them considered the sensitive attribute, not used for the optimization. The second dataset corresponds to one of the most widely used by the *Algorithmic Fairness* community, a criminal recidivism prediction dataset composed by more than 6000 samples, with 18 features describing each individuals. For each dataset, we consider both the white-box and the black-box attack scenarios described in Section 2.3.

3.1 Experiments with synthetic data

The first round of experiments uses synthetic data set to empirically test the impact of the attacks with respect to varying levels of disparity already found in the (unaltered) training data. Data is generated using the same approach of Zafar et al. [25]. Specifically, we generate 2,000 samples and assign them to binary class labels ($y = +1$ or $y = -1$) uniformly at random. Each sample is represented by a 2-dimensional feature vector created by drawing samples from two different Gaussian distributions: $p(x|y = +1) \sim N([2; 2], [5, 1; 1, 5]$ and $p(x|y = -1) \sim N([\mu_1; \mu_2], [10, 1; 1, 3])$ where μ_1, μ_2 are used to modify the euclidean distance S between the centroids of the distributions for the privileged and unprivileged groups so that different base rates [7] can be tested in the experiments. Then, a sample's sensitive attribute z is assigned by drawing from a Bernoulli distribution using $p(z = +1) = \frac{p(x'|y=+1)}{p(x'|y=+1)+p(x'|y=-1)}$ where $x' = [cos(\phi) - sin(\phi); sin(\phi), cos(\phi)]x$ corresponds to a rotated version of the feature vector x.

Using the generator we have described, datasets such as the ones as depicted in Figure 3 can be obtained. In this figure, the feature vector x is represented in the horizontal and vertical axes, while the color represents the assigned label

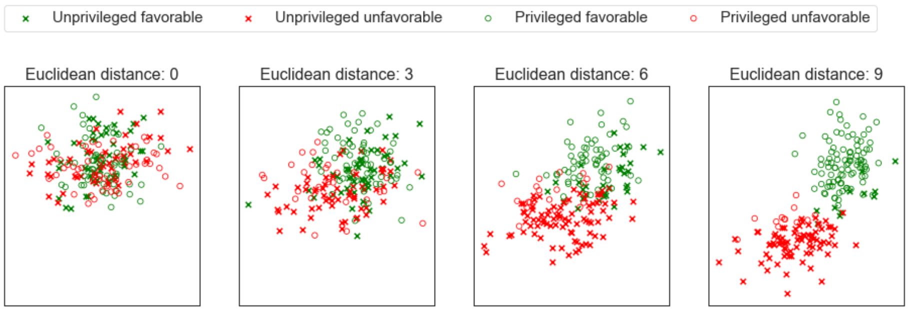

Fig. 3. (Best seen in color.) Examples of generated synthetic data sets for different values of the separation S between groups.Privileged elements ($z = +1$) are denoted by circles and unprivileged elements ($z = -1$) by crosses. Favorable labels ($y = +1$) are in green, while unfavorable labels ($y = -1$) are in red.

y (green means favorable, red means unfavorable) and the symbol the sensitive attribute z (circle means privileged, cross means unprivileged).

We generate multiple datasets by setting $S \in \{0, 1, 2, \ldots, 9\}$. We then split each dataset into training D_{tr} (50% of the samples), validation D_{val} (30%) and testing D_{test} (20%) subsets. In each run, a base or initial model $\mathcal{M}$ is trained. This model $\mathcal{M}$ corresponds to a Logistic Regression model in the first setting and to a Support Vector Machine with linear kernel in the second scenario. The regularization parameter C is automatically selected between $[0.5, 1, 5, 10]$ through cross validation. In the *White-Box* setting, the attack is optimized for $\mathcal{M}$ so that Eq. 1 is minimized in the training set D_{tr} and Eq. 3 is maximized in the validation set D_{val}. In the *Black-Box* setting, the attack is optimized against a surrogate model $\hat{\mathcal{M}}$, a Logistic Regression classifier, trained with another subset of data generated for the same value of the parameter S Each of these attacks generates a number of poisoning samples. The poisoned model is the result of retraining the original model with a training set that is the union of D_{tr} and the poisoned samples.

The attack performance is measured by comparing the model trained on the original training data with a model trained on the poisoned data. The evaluation is done according to the following metrics, which for each dataset are averaged over ten runs of each attack:

- **Accuracy** The accuracy on test obtained by the poisoned model is similar and correlated with the accuracy obtained by a model trained on the original data. It is important to note that the separability of the generated data is also highly correlated with the separation between the groups in the data, creating this effect.
- **Demographic parity** Measures the allocation of positive and negative classes across the population groups. Framed within the Disparate impact criteria that aims to equalize assigned outcomes across groups, this metric

is formulated as:

$$P(\hat{Y}=1|G=\mathit{unprivileged}) - P(\hat{Y}=1|G=\mathit{privileged})$$

It tends to zero in a fair scenario and is bounded between $[1,-1]$ being -1 the most unfair setting. This metric is correlated with the *Disparate impact* metric introduced in Section 2 and has been selected for convenience in the visual representation of the results.

- **Average odds difference** The average odds difference is a metric of disparate mistreatment, that attempts for Equalized odds [12], it accounts for differences in the performance of the model across groups. This metric is formulated as:

$$\frac{1}{2}[(FPR_p - FPR_u) + (TPR_p - TPR_u)]$$

  It gets value zero in a fair scenario and is bounded between $[1,-1]$ being -1 the most unfair setting.
- **FNR privileged** False Negative Rate for the privileged group of samples.
- **FNR unprivileged** False Negative Rate for the unprivileged group of samples.
- **FPR privileged** False Positive Rate for the unprivileged group of samples.
- **FPR unprivileged** False Positive Rate for the unprivileged group of samples.

Results shown on Figure 4 show the obtained performance of the attacks for the generated data. In this figure, the horizontal axis is the separation S between classes in each of the ten datasets. Analyzing the results, we observe that the poisoned models increase disparities in comparison with a model created on the unaltered input data, across all settings. Additionally, they yield an increased FPR for the privileged group (privileged samples that actually have an unfavorable outcome are predicted as having a favorable one), increasing significantly the observed unfairness as measured by the fairness measurements. We note that the attacks also decrease the FNR of the unprivileged group (unprivileged samples that actually have a favorable outcome are predicted as having an unfavorable one). This is most likely a consequence of the attack's objective of maintaining accuracy and show that this attack is not trivial. If the attack were only to increase disparities, it would also increase the FNR of the unprivileged group with a larger decrease in accuracy than what we observe. The decrease of FNR for the unprivileged group, however, is smaller than the increase of FPR for the privileged group, as the average odds difference plot shows, and hence the attack succeeds.

3.2 Experiments with real data

To demonstrate the attacks on real data, we use the COMPAS dataset released by ProPublica researchers [1], which is commonly used by researchers on Algorithmic Fairness. This dataset contains a prediction of criminal recidivism based

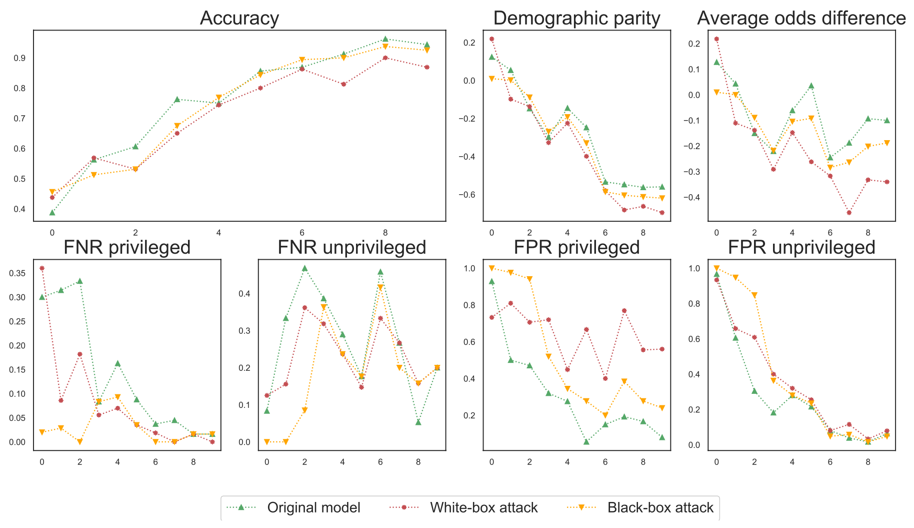

Fig. 4. Comparison of the original model against the model generated by the White-box attack and Black-box attacks, for ten synthetic datasets generated by different separation parameters (S). Each data point is the average of ten runs of an attack. We observe that attacks have a moderate effect on the accuracy of the classifier, and can affect the classifier fairness (demographic parity and odds difference) to an extent that becomes more pronounced if the original dataset already has a large separation between classes (larger values of S).

on a series of attributes for a sample of $6,167$ offenders in prison in Broward County, Florida, in the US. The attributes for each inmate include criminal history features such as the number of juvenile felonies and the charge degree of the current arrest, along with sensitive attributes: race and gender. For each individual, the outcome label ("recidivism") is a binary variable indicating whether he or she was rearrested for a new crime within two years of being released from jail.

We use this dataset for two different types of experiments. First, we show how the attacks demonstrated on synthetic data can also be applied to this data, and demonstrate the effect of varying the amount of poisoned samples, Second, we evaluate the transferability of the attack to other classification models.

White-Box and Black-Box poisoning attacks with varying amounts of poisoned samples. This experiment compares the original model against the model obtained under the two attack models.

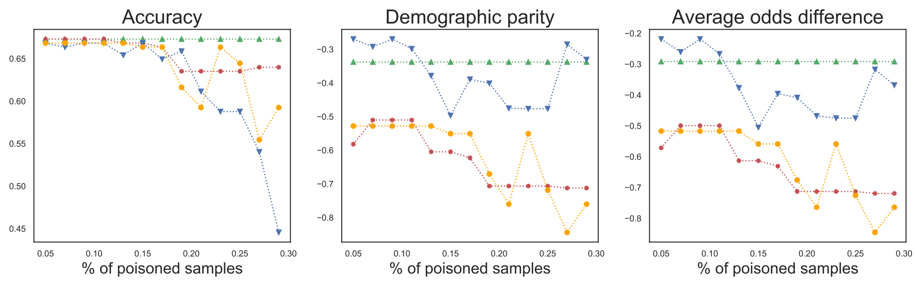

Fig. 5. Comparison of the original model against the model generated by a White-box attack and a Black-box attack, for varying percentages of poisoned samples. The main difference between both types of attack is that the black-box attack starts having more noisy behaviour also drastically reducing the accuracy of the classifier (thus being more easily detectable) when the percentage of poisoned samples exceeds a certain threshold (about 20%).

Figure 5 shows the results, which are in line with the findings of the experiments on synthetic data. According to the obtained results, both types of poisoning attacks are is able to increase unfairness of the model with a more modest effect on the accuracy. Also, an interesting finding is the stability of the *White-Box* attack as opposite to the *Black-Box* attack. Whereas the first keeps the same trend with the growing number of samples, the later starts having a unstable and noisy behaviour after adding the 20% of samples, causing for some cases a more unfair model but also affecting the accuracy of the system in a manner that could be easily detected.

In Figure 5 we also include an Error-Generic Poisoning Attack [8] for the Logistic Regression model , which is designed to decrease the accuracy of the resulting model. We observe that this type of generic adversarial machine learning attack does not affect the fairness of the classifier nearly as much as the attacks we have described on this paper.

As expected, computing the obtained performance for all the stated metrics, (Figure omitted for brevity) can be observed that the effect of any attack increases with the number of poisoned samples. In general, these attacks increase the False Negatives Rate (FNR) for the unprivileged samples, and increase the False Positives Rate (FPR) for the privileged samples.

Transferability of the attack. We study how an attack would affect the performance of other type of models, simulating different scenarios of *Zero Knowledge* attacks.

Specifically, the attacks we perform is optimized for a Logistic Regression model, and its performance is tested for other models: (a) Gaussian Naive Bayes. (b) Decision Tree; (c) Random Forest; (d) Support Vector Machine with linear kernel; and (e) Support Vector Machine with Radial Basis Function (RBF) kernel.

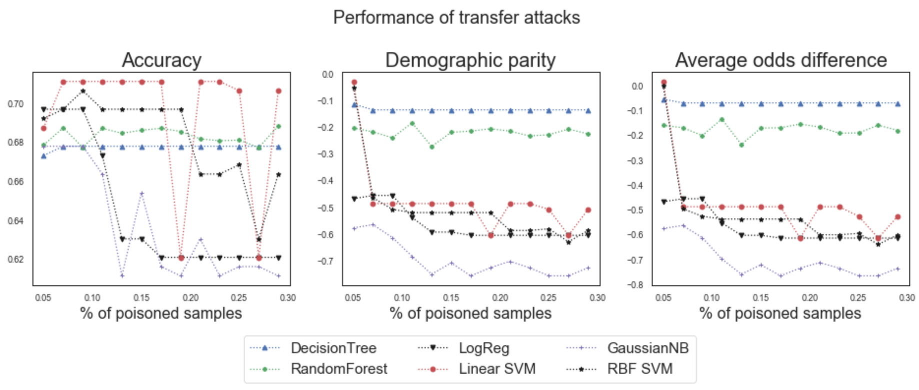

Fig. 6. Transferability of the attacks from Logistic Regression to other models.

Results are shown on Figure 6, in which each data point corresponds to the average of five experimental runs. We observe that the attack optimized on a Logistic Regression classifier has a stronger effect on the Logistic Regression, Support Vector Machine (for both types of kernel tested) and Naive Bayes models. In contrast, while it can introduce unfairness through demographic disparity and average odds difference on a Decision Tree or Random Forest classifier, its effects are more limited.

4 Related Work

Adversarial Machine Learning Attacks. This work is based on Gradient-Based Optimization, an optimization framework widely used in the literature on Adversarial Machine Learning for crafting poisoning attacks [5,15,17,13,8]. Such framework is used to solve the bilevel optimization given by Eqs. (1)-(3), and requires computing the gradient of the classification function learned by the classifier. As a result, poisoning samples can be obtained by iteratively optimizing one attack point at a time [22].

Measuring Algorithmic Fairness. Many different ways of measuring algorithmic fairness have been proposed [18]. Among those that can be applied in an automatic classification context we find two main types: individual fairness metrics and group fairness metrics [11]. The former seek *consistency* in the sense that similar elements should be assigned similar labels [9]. The latter seek some form of *parity*, and in many cases can be computed from a contingency table indicating the number of privileged and unprivileged samples receiving a positive or negative outcome [20]. Popular group fairness metrics include disparate impact, equalized odds [12], and disparate mistreatment [23].

Optimization-Based Approaches to Increase Fairness. Algorithmic fairness can and often is compromised unintentionally, as discrimination in machine learning is often the result of training data reflecting discriminatory practices that may not be apparent initially [2]. When this is the case, training data can be modified by a type of poisoning attack, in which so-called "antidote" samples are added to a training set to reduce some measure of unfairness. One such approach proposes a method to be applied on recommender systems based on matrix factorization [21]; another is based in the Gradient-Based Optimization framework used in this work [14].

In addition to methods to mitigate unfairness by modifying training data (something known as a pre-processing method for algorithmic fairness [11]), other methods modify the learning algorithm itself to create, for instance, a fair classifier [25,23] In these works, the trade-off between accuracy and fairness is approached through an alternative definition of fairness based in covariance between the users' sensitive attributes and the signed distance between the feature vectors of misclassified users and the classifier decision boundary.

5 Conclusions and Future Work

The results show the feasibility of a new kind of adversarial attack crafted with the objective of increasing disparate impact and disparate mistreatment at the level of the system predictions. We have demonstrated an attacker effectively alter the algorithmic fairness properties of a model even if pre-existing disparities are present in the training data. This means that these attacks can be used to both introduce algorithmic unfairness, as well as for increasing it where it already exists. This can be done even without access to the specific model being used, as a surrogate model can be used to mount a black-box transfer attack.

Studying adversarial attacks on algorithmic fairness can help to make machine learning systems more robust. Although experiments in this paper are done using a specific technique based on a poisoning attack, other techniques can be certainly considered. Other approaches such as causality-based techniques could be explored as future work.

6 Acknowledgements

This research was supported by the European Commission through the ALOHA-H2020 project. Also, we wish to acknowledge the usefulness of the Sec-ML library [16] for the execution of the experiments of this paper.

References

1. Angwin, J., L.J.M.S., Kirchner, L.: Machine bias. there's software used across the country to predict future criminals. and it's biased against blacks (2016)
2. Barocas, S., Hardt, M.: Fairness in machine learning nips 2017 tutorial (2017), `https://mrtz.org/nips17/#/`
3. Barocas, S., Selbst, A.D.: Big data's disparate impact. Calif. L. Rev. **104**, 671 (2016)

4. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
5. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Langford, J., Pineau, J. (eds.) 29th Int'l Conf. on Machine Learning. pp. 1807–1814. Omnipress (2012)
6. Buolamwini, J., Gebru, T.: Gender shades: Intersectional accuracy disparities in commercial gender classification. In: Proceedings of the 1st Conference on Fairness, Accountability and Transparency. pp. 77–91 (2018)
7. Chouldechova, A.: Fair prediction with disparate impact: A study of bias in recidivism prediction instruments (2016)
8. Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., Roli, F.: Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In: 28th USENIX Security Symposium (USENIX Security 19). USENIX Association (2019)
9. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: Proceedings of the 3rd innovations in theoretical computer science conference. pp. 214–226 (2012)
10. Feldman, M., Friedler, S., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact (2015)
11. Hajian, S., Bonchi, F., Castillo, C.: Algorithmic bias: From discrimination discovery to fairness-aware data mining. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 2125–2126 (2016)
12. Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning (2016)
13. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: Poisoning attacks and countermeasures for regression learning (2018)
14. Kulynych, B., Overdorf, R., Troncoso, C., Gürses, S.: Pots: Protective optimization technologies. Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (Jan 2020). https://doi.org/10.1145/3351095.3372853, `http://dx.doi.org/10.1145/3351095.3372853`
15. Mei, S., Zhu, X.: Using machine teaching to identify optimal training-set attacks on machine learners. In: 29th AAAI Conf. Artificial Intelligence (AAAI '15) (2015)
16. Melis, M., Demontis, A., Pintor, M., Sotgiu, A., Biggio, B.: secml: A python library for secure and explainable machine learning (2019)
17. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: Thuraisingham, B.M., Biggio, B., Freeman, D.M., Miller, B., Sinha, A. (eds.) 10th ACM Workshop on Artificial Intelligence and Security. pp. 27–38. AISec '17, ACM, New York, NY, USA (2017)
18. Narayanan, A.: Translation tutorial: 21 fairness definitions and their politics. In: Proc. Conf. Fairness Accountability Transp., New York, USA (2018)
19. Papernot, N., McDaniel, P.D., Goodfellow, I.J.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. ArXiv e-prints **abs/1605.07277** (2016)
20. Pedreschi, D., Ruggieri, S., Turini, F.: A study of top-k measures for discrimination discovery. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 126–131. SAC '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2245276.2245303, `http://doi.acm.org/10.1145/2245276.2245303`

21. Rastegarpanah, B., Gummadi, K.P., Crovella, M.: Fighting fire with fire. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining - WSDM '19 (2019). https://doi.org/10.1145/3289600.3291002, http://dx.doi.org/10.1145/3289600.3291002
22. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? In: Bach, F., Blei, D. (eds.) JMLR W&CP - Proc. 32nd Int'l Conf. Mach. Learning (ICML). vol. 37, pp. 1689–1698 (2015)
23. Zafar, M.B., Valera, I., Gomez Rodriguez, M., Gummadi, K.P.: Fairness beyond disparate treatment & disparate impact. Proceedings of the 26th International Conference on World Wide Web - WWW '17 (2017)
24. Zafar, M.B., Valera, I., Rodriguez, M., Gummadi, K., Weller, A.: From parity to preference-based notions of fairness in classification. In: Advances in Neural Information Processing Systems. pp. 229–239 (2017)
25. Zafar, M.B., Valera, I., Rodriguez, M.G., Gummadi, K.P.: Fairness constraints: Mechanisms for fair classification (2015)