Attack for Good: Black-box Data Poisoning for Fairness

Zhenhao Chen, William de Vazelhes, Boyang Sun
Mohamed bin Zayed University of Artificial Intelligence
Abu Dhabi, United Arab Emirates
{Firstname.Lastname}@mbzuai.ac.ae

1 Background

Algorithmic fairness is an imperative metric of machine learning (ML) to prevent potential discrimination among various applications. However, AI models show vulnerability towards bias, which may cause destructive effects without correction when applied to tasks like loan applications [1], dating and hiring [2, 3]. Beyond that, optimizing methods are usually limited due to tight restrictions on the access of user data and model because of privacy regulations. For example, giant companies will train the prediction model based on their users and products information but cannot manipulate the training dataset directly. Under these circumstances, a method of improving model fairness without touching the original training dataset is required. Data poisoning (see section 2.2) is a very efficient way to do so, because many actual systems deployed in real-life use web scrapers to build their training sets. Therefore, an attacker will only need to devise poisonous samples that will automatically be absorbed by the company's scrapper, and subsequently influence the prediction of that model. In this paper, we study poisonous attacks that will increase a model fairness. We formulate our problem as a *bilevel optimization* problem (see section 2.3). In addition, since the model we wish to attack is considered unknown, we work under the *black-box optimization* framework (see section 2.4) using *bayesian optimization* (see section 2.4.1). Our code is available at `https://github.com/viewsetting/Attack4Good`.

2 Related works

2.1 Fairness

Fairness is an increasingly popular topic of interest in the machine learning community [4]. The general idea of fairness is to ensure that a model will not make predictions that put one group or another to an extra advantage. This general definition can be refined into many ways, which leads to different quantitative metrics for fairness. Here we describe a metric for fairness that possess several attractive properties: the *equality of opportunity* metric [5]. For example in the task of evaluating whether a criminal will recidive or not, this metric ensures that if a person is not a recidivist, he/she will have the same probability to be classified as recidivist, no matter the value of a given sensitive attribute (e.g. group ethnicity). In other words, every innocent, no matter its characteristics, should have the same probability to be misclassified as criminal. More precisely, we define the following *equality of opportunity* metric $EO(w, \mathcal{D}_{\text{val}})$ computed for the validation set $\mathcal{D}_{\text{val}}$ and for the model w, that can be computed for instance on a held-out validation data set $\mathcal{D}_{\text{val}}$ after training.

$$EO(w, \mathcal{D}_{\text{val}}) = |FPR(w, \mathcal{D}_{\text{val}}[s = 1]) - FPR(w, \mathcal{D}_{\text{val}}[s = 0])| \tag{1}$$

Where s represent a sensitive attribute, $\mathcal{D}_{\text{val}}[s = 1]$ stands for the validation set *restricted to the individuals that have the value* 1 *for their senstiive attribute*, and $FPR(\mathcal{D})$ stands for the False Positive Rate on dataset $\mathcal{D}$, defined as follows:

$$FPR(w, \mathcal{D}) = \frac{FP(w, \mathcal{D})}{FP(w, \mathcal{D}) + TN(w, \mathcal{D})} \tag{2}$$

With $TP(w, \mathcal{D})$ being the number of true positive for the model w computed on dataset $\mathcal{D}$, that is the number of positive samples correctly classified as positive by the model w, and $FP(w, \mathcal{D})$ being the number of false positives, that is the number of negative samples incorrectly classified as positive.

Other metrics for fairness also exist, like the *equalized odds* metric, that ensures that members from both groups have the same probability to be classified as positive. The advantage of equality of opportunity over equalized odds is that it is less in contradiction with accuracy: indeed, equalized odds may force a classifier to predict a positive class on some group, even if the true underlying distribution is one with very few positives for this group. Equality of opportunity on the other hand, tries to make as much *mistakes* on both groups, but there is no additional absolute constraints on how many positives or negatives it should predict for both groups, as described in [5].

2.2 Data Poisoning

Data poisoning ([6], [7], [8]) is an increasingly popular field in machine learning, which studies how one can modify a model just by changing its training set. In this setting, an attacker tries to come up with some samples such that, when added to the training set of the model, they will modify the model. The goal of the attacker is for instance to harm the final accuracy of the model. In the next subsection, we discuss works that actually target the fairness of the model.

2.2.1 Data Poisoning Targeting Fairness

Perhaps the work the most similar to ours is the work from [9]. In that work, they study how data poisoning can actually harm fairness. They provide two possible formulations for their attacks. The first one is white-box attacks, in which the attacker has full access to the attacked model. In this case, they use gradient-based bi-level optimization (see section 2.3) to search for the optimal poisonous samples. The second one is black-box attacks, in which the attacker does not have access to full model. In that case however, the attacker will fall back into the previous strategy and do a white-box attack on his own model, which is called a *surrogate* model, and that is supposed to mimick the true, unknown model. Then the attackers uses the optimized poisonous samples for that true model, in the hope that they will *transfer* well to it. In practice, if the surrogate model is well chosen, the poisonous samples will indeed transfer well to the true model [10], [11].

The first difference with our work is that we want to devise poisonous examples that *increase*, not *decrease* the fairness. But this difference is very minor, since it is just a change of perspective. The main difference is that in our work, instead of gradient-based bi-level optimization with a surrogate model, we rather explore Bayesian optimization (see section 2.4.1) to search for the poisonous samples. The advantage is that we do not need a surrogate model anymore, we can directly optimize the poisonous samples by just observing the fairness score, after each training on the poisoned training set. To further test the feasibility of gradient-based optimization like [9], we also implement the similar method without getting ideal result. One can find it in the appendix.

We also note the work from [12], in which they also consider some poisoning attacks on fairness. However, they consider two kind of attacks, different from our framework: anchoring attacks, in which they place the poisonous samples close to some specific train-set points to skew the decision region of the classifier, and influence attacks, which tries to augment the correlation between sensitive attributes and decision outcomes.

2.3 Bi-level Optimization

Bi-level optimization [13] is a field of optimization that allows to optimize functions of a particular form, where the objective can be decomposed into two nested problems. It can be used for various problems in machine learning, from hyperparameter search ([14], [15]), data denoising [16], few-shot learning ([17]), or data poisoning ([9], [6]). A general bi-level optimization problem can be

formulated as follows, where λ is the variable we want to optimize, θ is the variable that is optimized by the inner optimization problem, with θ^* being the optimal value for that problem, and f and g are two functions.

$$\arg\min_{\lambda} g(\theta^*, \lambda) \text{ s.t. } \theta^* \in \arg\min_{\theta} f(\theta, \lambda) \tag{3}$$

This is much more challenging to optimize than a single level optimization problem, and several methods have been developped to tackle it, for example using gradient based ([8]), zeroth-order ([14]), or Bayesian ([18]) optimization.

2.4 Black-Box Optimization

Black-box optimization relates to any optimization method that does not have access to the gradient of the model to optimize. In the following section, we describe a very popular method for black-box optimization: Bayesian optimization.

2.4.1 Bayesian Optimization

Bayesian Optimization (BO) [19] is a very general method to optimize any function, by only observing its outputs for given inputs. It works by starting with a *prior belief* on the function to optimize, that is updated using the Bayes rule as more and more observations are sampled by the BO algorithm, hence its name Bayesian optimization. In order to sample the fewest observations as possible (since those can be costly), the BO algorithm at the same times need to refine its modeling of the function by sampling in uncertain regions of the parameter space (called *exploration*), and to *exploit* this built up confidence by actually focusing on regions where the objective function is high (for a maximization problem). This leads to a trade-off between exploration and exploitation for when to sample next, and in BO this trade-off is typically dealt with by the *acquisition function*.

Regarding the choice of the probability distribution modeling the considered function, a typical choice is to use Gaussian processes, that model the uncertainty at a particular a particular point of the parameter space by a Gaussian distribution.

2.5 Gaussian processes

In this subsection, we describe the probability distribution used to model our function to optimize, namely, Gaussian Processes [20]. Gaussian Processes model the probability of each function value $f(x)$, conditioned on the previous ones, with a Gaussian probability, as follows:

$$f(x) \mid (f(x_1) = y_1, f(x_2) = y_2, \cdots, f(x_N) = y_N) \sim \mathcal{N}\left(\mathbf{k}^T \Sigma^{-1} y, K(x,x) - \mathbf{k}^T \Sigma^{-1} \mathbf{k}\right)$$

Where

$$\Sigma = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & \cdots & K(x_N, x_N) \end{bmatrix}$$

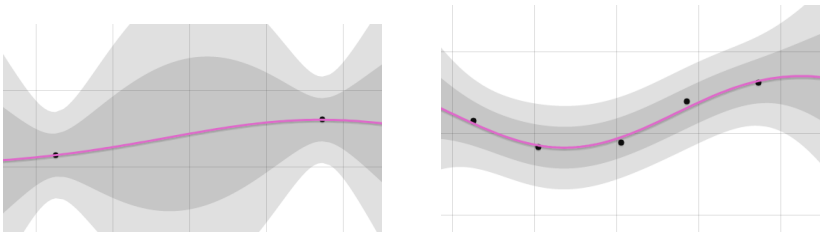
and

$$K(x, y) = e^{(\frac{-||x-y||^2}{2l^2})}$$

where l is the length scale of the kernel. Here, K is the RBF (Radial Basis Function) kernel, defined as follows:

$$\mathbf{k} = \begin{bmatrix} K(x_1, x) & K(x_2, x) & \cdots & K(x_N, x) \end{bmatrix}^T$$

In fact, when N samples have been drawn, they define a prior distribution, and the formula above describes that when sampling the $N+1$-th sample, we will update that prior into a posterior distribution.



(a) Prior (2 points observed) (b) Posterior (5 points observed)

Figure 1: Gaussian process (source: http://chifeng.scripts.mit.edu/stuff/gp-demo/)

This means we start with a lot of uncertainty about the function f, but after a few observations, the uncertainty about the function f diminishes, like explained in figure 1.

2.6 Acquisition function

The BO algorithm needs to deal with this uncertainty to sample as few points as possible to find the optimum of the function. In order to do that, the algorithm needs to *exploit* its knowledge of the function, and sample points in regions of low value functions (for a minimization problem), but also *explore* the parameter space, and ensure it is not missing some potentially good regions in the uncertain parts of the space. This leads to the so-called *exploration-exploitation* dilemma.

As mentioned above, the sampling strategy of the BO algorithm is dealt with by the acquisition function. In our case, we use the *Upper Confidence Bound* technique [21]: the value of the acquisition function at a point x, with mean and variance of the Gaussian Process distribution being $\mu(x)$ and $\sigma(x)^2$ respectively (see 2.5) for the way those are computed), is defined as follows:

$$a(x) = \mu(x) + \kappa\sigma(x)$$

where κ is a parameter to choose, that trades off exploitation and exploration (see below).

At every timestep, the BO algorithm finds the x that maximizes this activation $a(x)$, then observes the function value for it $f(x)$, updates its prior distribution on f into a posterior, using the tuple $(x, f(x))$, and starts the procedure of acquisition function maximization again.

Therefore, the higher the mean $\mu(x)$, the more likely we are to select that x. This is useful for exploitation as it allows us to focus on the points of higher value. However, we can also select some x if the variance there is high. This way, it allows us to also *explore* regions of the parameter space where the function values *could* possibly be high. Therefore, higher values for κ allow us to do more exploration. Ways to find the optimal value for κ are for instance explained in [21].

3 Method Description

3.1 Contribution

Our method combines the methods exposed above, namely black-box optimization (section 2.4), and data-poisoning for fairness (section 2.2.1). Our contribution is to provide the machine learning community with a framework to inject poisonous examples in the training set, such that the fairness of the trained model will be improved. Contrary to [9], we devise the poisonous samples in a fully black-box fashion, that is, without having access to the model or its gradients.

3.2 Optimization Problem

Our general formulation can be described as follows: we wish to optimize the poisonous samples $\mathcal{D}_p$, such that the fairness metric $\mathcal{F}(\mathcal{D}_{\text{valid}}, w^*)$ on the trained model w^* evaluated on the validation set $\mathcal{D}_{\text{valid}}$, is maximized. As an example for the fairness metric $\mathcal{F}(\mathcal{D}_{\text{valid}}, w^*)$, one could choose the opposite of the equality of opportunity metric, defined in 1 (we take the opposite of that metric, since we want to minimize the difference of false positive rates between the two groups).

The trained model w^* is a minimizer of a loss function $\mathcal{L}$, computed on a training set $\mathcal{D}_{\text{train}}$. This latter minimization is called the *inner problem*, and can be for instance the minimization of the empirical mean cross-entropy computed on the training set. Therefore, we have the following mathematical formulation:

$$\max_{\mathcal{D}_p} \mathcal{F}(\mathcal{D}_{\text{valid}}, w^*) \quad s.t.\ w^* \in \arg\min_{w} \mathcal{L}(\mathcal{D}_{\text{train}} \cup \mathcal{D}_p, w) \tag{4}$$

3.3 Bayesian Optimization

The method we use to solve the optimization problem defined in 5 is using Bayesian optimization, discussed in 2.4.1. We use vanilla bayesian optimization (BO) to search for the best $\mathcal{D}_p$. BO updates the posterior of the gaussian process then uses its aquisition function to estimate the fairness related metric of the observation poison point $\mathcal{F}_{\text{observe}}$. We keep BO estimating and observing new samples until it finds one poison sample that can improve model's $\mathcal{F}$ on valid dataset.

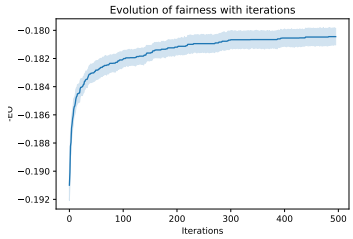
Algorithm 1 Vanilla Bayesian Optimization (BO)

Require: : Current max value of fairness metric $\mathcal{F}_{\text{max}}$, weights of model w
  $\mathcal{F}_{\text{observe}} \leftarrow \mathcal{F}_{\text{max}}$
  $counter \leftarrow 0$
  while $\mathcal{F}_{\text{max}} \geq \mathcal{F}_{\text{observe}}$ **do**
    Update the posterior distribution of $\mathcal{D}_p$ on $\mathcal{F}$ using all available data
    $\mathcal{D}_p^* \leftarrow \arg\max \text{aquisition function}(\mathcal{D}_p)$
    $w^* \leftarrow \arg\min_w \mathcal{L}(\mathcal{D}_{\text{train}} \cup \mathcal{D}_p^*, w)$
    $\mathcal{F}_{\text{observe}} \leftarrow \mathcal{F}(\mathcal{D}_{\text{valid}}, w^*)$
    $counter = counter + 1$
  end while
  return $\mathcal{D}_p^*$, $counter$, w^*

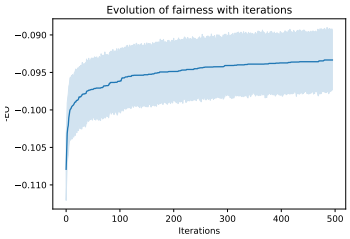
In algorithm (1), if given $\mathcal{F}_{\text{max}}$, BO will try to find one poison data $\mathcal{D}_p^*$ which could increase model's fairness metric on valid dataset. However, BO is time-consuming compared with gradient-like method. So in practice, we set a computation budget limit for it. Because Gaussian Process needs some random points to build its kernel matrix and the number of points is t_s. That means BO has a number of $T - t_s$ of trials to find poison data as much as possible. We summarize the algorithm in the pseudocode below:

Algorithm 2 Iterative Poison Attack

Require: Number of iterations (budget): T,
  number of initial random samples for BO: t_s
  Train model on $\mathcal{D}_{\text{train}}$ and update w
  Use t_s randomly selected samples to initialize BO
  $\mathcal{F}_{\text{max}} \leftarrow \mathcal{F}(\mathcal{D}_{\text{valid}}, w)$
  $\mathcal{D}_p \leftarrow \{\}$
  for $t = t_s + 1 : T$ **do**
    $\mathcal{D}_p^*, counter, w \leftarrow BO(\mathcal{F}_{\text{max}}, w)$
    $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}_p^*$
    $t \leftarrow t + counter$
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup \mathcal{D}_p^*$
  end for
  return w, $\mathcal{D}_p$

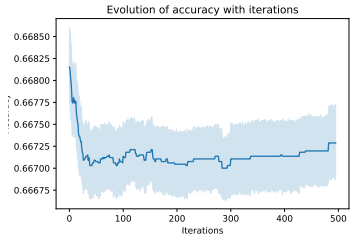


(a) Gradient Boosting Classifier

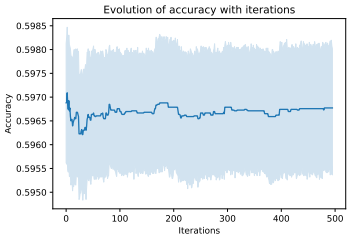


(b) Random Forest Classifier

Figure 2: Evolution of fairness along training



(a) Gradient Boosting Classifier



(b) Random Forest Classifier

Figure 3: Evolution of accuracy along training

4 Experiments

4.1 Dataset

We make our experiments on the COMPAS dataset [22]. We download that dataset from the OpenML repository [23]. This dataset gathers data from criminals in the Broward County in the US, to try predict whether they will be a recidivist or not. It is composed of 5,278 rows samples of 13 attributes each, that comprise variables like the number of times the individual has commited a juvenile felony, their age category, as well as sensitive attribute like gender and ethnic group. For each person, the label is a binary variable that indicates whether or not they has recidived within two years after their release from jail.

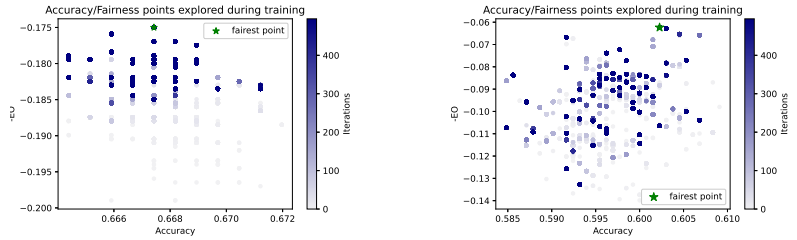
4.2 Experiment Setting

We run the BO with random forest and gradient boosting tree for the inner loop, optimized with cross-entropy. For each experiment, we give a total computational budget of 500. For initialization of BO, we randomly select 5 points. That makes BO has the limit of 495 trials for observation. For each model, we run algorithm (2) for 50 times with different random seeds respectively.

4.3 Results

Figure (2) shows that equal opportunity of both random forest and gradient boosting classifier increase during poison attack. The variance of fairness of gradient boosting is clearly lower than random forest's. According to figure (3), along training, the accuracy of both gradient boosting and random forest nearly doesn't drop (only less than 0.1 percent in gradient boosting classifier). The improvement of fairness is much larger in random forest, although fairness performance of it is quite better than gradient boosting before poinsoning. For gradient boosting classifier, in the best situation fairness can be improved by 12.5 percent and for random forest it can be up to 53.4 percent.

Discussion The bayesian optimization(BO) based poison attack can improve machine learning model's performance measured by fairness related metric in a black-box way. This method doesn't



(a) Gradient Boosting Classifier (b) Random Forest Classifier

Figure 4: Accuracy/Fairness points discovered

need access to any information of model and even input data which makes privacy preserving naturally achieved. Although BO has many limitations like more time-consuming, hard to converge in high dimension etc, it has nearly no constraint on convexity and differentiability of optimization problem. That makes BO as the preferred choice for black-box poison attack more generally adapted to different scenarios. However, those limitations are worthy to tackle for better extension to more complex datasets with numerous features.

References

[1] Amitabha Mukerjee, Rita Biswas, Kalyanmoy Deb, and Amrit P Mathur. Multi–objective evolutionary algorithms for the risk–return trade–off in bank loan management. *International Transactions in operational research*, 9(5):583–597, 2002.

[2] Miranda Bogen and Aaron Rieke. Help wanted: An examination of hiring algorithms, equity, and bias. 2018.

[3] Lee Cohen, Zachary C Lipton, and Yishay Mansour. Efficient candidate screening under multiple tests and implications for fairness. *arXiv preprint arXiv:1905.11361*, 2019.

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

[5] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323, 2016.

[6] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[7] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[8] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[9] David Solans, Battista Biggio, and Carlos Castillo. Poisoning attacks on algorithmic fairness. *CoRR*, abs/2004.07401, 2020.

[10] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[11] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 321–338, 2019.

[12] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. Exacerbating algorithmic bias through fairness attacks. *CoRR*, abs/2012.08723, 2020.

[13] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[14] Bin Gu, Guodong Liu, Yanfu Zhang, Xiang Geng, and Heng Huang. Optimizing large-scale hyperparameters via automated learning algorithm. *CoRR*, abs/2102.09026, 2021.

[15] Wanli Shi, Bin Gu, and Heng Huang. Improved penalty method via doubly stochastic gradients for bilevel hyperparameter optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9621–9629, 2021.

[16] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[18] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Bayesian optimization approach of general bi-level problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1614–1621, 2017.

[19] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[20] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[21] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[22] Kirchner Larson, Mattu and J. Angwin. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks., 2016.

[23] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[24] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

Appendix A Gradient-based Optimization

We follow the method in [9] which implemented fairness attack through gradient-based update way. By changing the optimization objective and constraint, we aimed at improving the fairness of the trained classifier. We reformulate equation(4) as (5) with relating the fairness metric $F(D_{valid}, w_*)$ with a differentiable loss $L(D_{valid}, w_*)$, which can be expressed as a function $A(x_c, y_c)$ whose variable is the injected data.

$$\min_{x_c} A(x_c, y_c) = L(D_{val}, w^*) \quad s.t.\ w^* \in \arg\min_{w} \mathcal{L}(\mathcal{D}_{\text{train}} \cup (x_c, y_c), w) \tag{5}$$

Now, the problem transfers to find out a differentiable loss which can reflect the attributes of fairness. Following the way of [9], we consider disparate impact criterion[24] whose intuition is to avoid the out of balance for the ratio of privileged and non-privileged with positive results. We can mathematically formulate it as:

$$D = \frac{P(\hat{y} = 1|G = u)}{P(\hat{y} = 1|G = p)} \geq \epsilon \quad s.t. \quad \epsilon \leq 1 \tag{6}$$

where u and p represent the unprivileged group and privileged groups respectively, $\hat{Y}$ means the predicted value. As we are enhancing the fairness of the classifier, the value of D should be as close to one as possible. As the numerator of D can't be larger than the denominator, Our optimization should make the numerator as large as possible while reducing the denominator until it is close to

the numerator. In other words, we are minimizing the difference between the ratio of privileged and unprivileged with prediction $\hat{Y} = 1$:

$$L(D_{val}, w^*) = \sum_{i=1}^{n} l(x_i, y_i \leftarrow 1, w^*) + \lambda \sum_{j=1}^{m} l(x_j, y_j \leftarrow 0, w^*) \tag{7}$$

where index i represents one sample in unprivileged group; j is the sample in privileged group; $\lambda = \frac{n}{m}$ to balance the contribution to loss of each group. The loss will only happens when $\hat{y} \neq y$. If we assign the label of unprivileged group to be 1, the label of privileged group to be 0, the loss will represent the condition when prediction of unprivileged group being 0, privileged group being 1. Minimizing this loss will force the classifier to increase numerator of D and decrease the denominator of D respectively, thus improving the fairness of the trained classifier.

Algorithm 3 Gradient-based optimization

$x_c = x_c^*, y_c :$ the injected data and the label
$\eta :$ the stepsize
$\tau :$ the convergence boundary
repeat
  $x_c \leftarrow x_c^*$
  ${x_c}^* \leftarrow x_c + \eta \nabla_{x_c} A$
until $|A(x_c^*, y_c) - A(x_c, y_c)| \leq \tau$
return x_c^*

The problems remains at the calculation of the gradient $\nabla_{x_c} A$ whose variable x_c and objective A is directly and implicitly related. In this case, we assume the inner objective function $\mathcal{L}(\mathcal{D}_{\text{train}} \cup (x_c, y_c), w)$ is smooth and its second derivative $\mathcal{L}$ is invertible. From the chain rule and implicit equation, the gradient can be calculated as

$$\nabla_{x_c} A = \nabla_{x_c} L - (\nabla_{x_c} \nabla_w \mathcal{L})(\nabla^2 \mathcal{L})^{-1} \nabla_w L \tag{8}$$

We operate this gradient using both white-box and black-box attack strategy. Interestingly, only black box works; the white-box injection even harms the fairness. The following figure shows the result. Where AOD and DP represent two metrics of fairness. The closer those two metrics are to 0, the fairer.

Appendix B Work division

We equally divided the tasks between us, in the following way:

- **Zhenhao Chen:** Full experiments with Bayesian optimization, writing of the Experiments section on Bayesian Optimization, some modifications in the manuscript, review of the final manuscript
- **Boyang Sun:** Full experiments with the method of ([9]), writing of the Experiments section on it, some modifications in the manuscript, review of the final manuscript
- **William de Vazelhes:** Minimal running example on a toy problem (using some existing code for BO), writing of a first draft of the manuscript (except experiments section), some modifications in the manuscript, review of the final manuscript

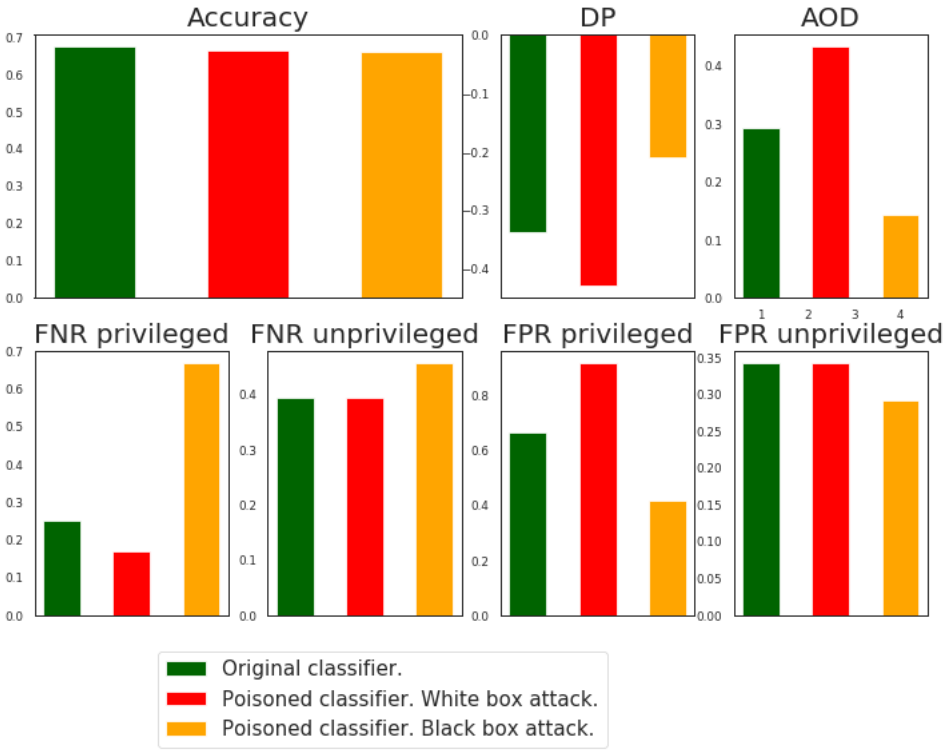


Figure 5: Result of gradient-based optimization