Redactor: Targeted Disinformation Generation using Probabilistic Decision Boundaries

Geon Heo  
KAIST  
Daejeon, Republic of Korea  
geon.heo@kaist.ac.kr

Steven Euijong Whang  
KAIST  
Daejeon, Republic of Korea  
swhang@kaist.ac.kr

ABSTRACT

Information leakage is becoming a critical problem as various information becomes publicly available by mistake, and machine learning models train on that data to provide services. As a result, one’s private information could easily be memorized by such trained models. Unfortunately, deleting information is out of the question as the data is already exposed to the Web or third-party platforms. Moreover, we cannot necessarily control the labeling process and the model training by other parties either. In this setting, we study the problem of targeted disinformation where the goal is to lower the accuracy of inference attacks on a specific target (e.g., a person’s profile) only using data insertion. While our problem is related to data privacy and defenses against exploratory attacks, our techniques are inspired by targeted data poisoning attacks with some key differences. We show that our problem is best solved by finding the closest points to the target in the input space that will be labeled as a different class. Since we do not control the labeling process, we instead conservatively estimate the labels probabilistically by combining decision boundaries of multiple classifiers using data programming techniques. We also propose techniques for making the disinformation realistic. Our experiments show that a probabilistic decision boundary can be a good proxy for labelers, and that our approach outperforms other targeted poisoning methods when using end-to-end training on real datasets.

1 INTRODUCTION

Information leakage is becoming a serious problem as personal data is being used to train machine learning (ML) models. As a striking example, Lee Luda [3] is an AI chat bot service that was shut down soon after its release because of its hate speech towards minorities and exposure of personal data including bank accounts and addresses. Another example is the Web [2] where life insurers can predict the life spans of their customers based on personal information leaked on the Web through various online activities. Recently, Flickr [1] has been in the news where photos uploaded in its early days are now used as training data for various ML computer vision models without author consent. Furthermore, there are various privacy threats on ML models including inference attacks [12, 26, 45] and reconstruction attacks [18, 19].

Unfortunately, one cannot simply delete the information if it is published on the Web or uploaded on a third-party platform, but can only dilute it. Even if the original data is deleted by request, there is no way to prevent someone from extracting that information elsewhere by attacking the model of the unknown third-party platform. Moreover, there is also no control over the model training process where anyone can train a model on the publicized data. Hence, conventional privacy techniques or inference attack defenses that require ownership of the data or model are out of the question. The only solution is to add more data to dilute one’s information and reduce the model’s accuracy and confidence of predictions on a certain target (e.g., personal profile) as shown in Figure 1. An analogy is blacking out or redacting text where the reader knows there is some information, but cannot read the text. We refer to this problem as targeted disinformation.

The most relevant work to disinformation generation is targeted poisoning [30, 44, 48, 53]. Recent techniques have been proposed mainly for attacking image classification models. Given a target image \( t \) and a base image \( b \) of a different class, the goal is to generate an image \( p \) that looks like \( b \) according to a labeler, but is classified the same as \( t \) by the model. If transfer learning is used, this optimization can be solved using a pre-trained model that is used to generate features of images. Then \( p \) can be made to be close to \( b \) in the input
space (i.e., the pixels are similar), but close to $t$ in the feature space. The labeler would label $p$ to be the same as $b$, but this will confuse the model training, which thinks that $p$ is the same as $t$. We would like to utilize such techniques to change the output of the unknown models (e.g., third-party platforms) and protect target instances from indirect privacy attacks. However, most of these techniques assume a transfer learning [33] scenario where a pre-trained model is used to generate features in a fixed feature space. While transfer learning benefits certain applications (e.g., NLP or vision tasks), it is not always applicable, especially for structured data where there is no efficient and generally accepted practice [7]. However, structured data is important because most personal information is stored in this format. In order to support structured data, we need to assume end-to-end training where the feature space is no longer fixed. As a result, existing targeted poisoning techniques cannot be used as is for our purposes.

While our approach is inspired by targeted poisoning attacks, the key difference is that we only utilize the input space to find the best disinformation that is close to the target, but labeled differently. How do we know the true label of the disinformation? Since we do not have access to the labelers, our key idea is to conservatively estimate human behavior using probabilistic decision boundaries produced by combining multiple possible classifiers. We adapt data programming [39–41], which combines multiple labeling functions into a label model that produces probabilistic labels. In our setting, we make the generative model produce the probability of an example having a class that is different than $t$’s class. By limiting this probability to be above a tolerance threshold, we now have a conservative decision boundary. To illustrate our approach, Table 1 shows people records with attributes Education and Age. Let us say $t$ is the target, and $b_1$ and $b_2$ are other examples. There are three binary surrogate models that produce the predictions $\hat{Y}_1$, $\hat{Y}_2$, and $\hat{Y}_3$. Let us say the probabilistic decision boundary labels an example as 0 (i.e., not in $t$’s class) only if at least 60% (intentionally low for illustration purposes) of the surrogate models say so. As a result, we can only safely say that $b_2$ is not the same person as $t$. Given this decision boundary, we can generate a disinformation $d_2$ that is labeled the same as $b_2$, but has the closest distance to $t$ as shown in Table 1 and Figure 2. We generate $d_2$ between $t$ and $b_2$ using watermarking [10, 44] where a watermark of $t$ is added to $b_2$ to generate $d_2$ using linear interpolation.

Our proposed system Redactor generates disinformation using probabilistic decision boundaries and base examples as shown in Table 1 and Figure 2. In addition, Redactor uses generative adversarial networks [23] to produce more disinformation examples like $d_3$ and $d_4$. Here $d_4$ happens to be even closer to $t$ than $d_2$. Finally, Redactor ensures the disinformation is realistic by avoiding patterns of data that do not occur in the original data.

We evaluate Redactor on real datasets using end-to-end model training. We show how the probabilistic decision boundaries are valid proxies of labelers and how Redactor outperforms other targeted poisoning attack baselines in terms of reducing the model’s accuracy and confidence on the target and thus making the model’s output more likely to change.

In the rest of the paper, we present the following:

- Overview of Redactor (Section 3) and its main components
- Probabilistic decision boundaries (Section 3.3).
- Disinformation generation (Section 3.4).
- Overall algorithm (Section 3.5).
- Experimental results (Section 4) where Redactor outperforms existing targeted poisoning approaches.
- Related work (Section 5).

## 2 BACKGROUND

We clarify the difference between a transfer learning scenario and end-to-end training scenario. We then explain why existing targeted poisoning attacks, which rely on transfer learning, fail to perform well in an end-to-end setting.

Comparison with Targeted Poisoning. Targeted poisoning attacks have the goal of flipping the predictions on specific targets to a certain class. A naive approach is to add examples that are identical to the target, but with different labels. Unfortunately, such an approach would not work if one does not have complete control over the labeling process, which is unrealistic. Instead, the poison $p$ needs to be different enough from the target to be labeled differently by any human. Yet, we also want $p$ to be close to the target as well.
The state-of-the-art targeted poisoning attacks include Convex Polytope Attack (CPA) [53] and its predecessors [10, 44, 48], which also do not assume any control over the labeling and generate poison examples that are similar to the base examples, but have the same predictions as the target. Like our setting, these techniques are not involved in the model training itself, but generate poisoned examples that are presumably added to the training set. The goal is to generate examples close to the target in the feature space while being close to a base example in the input space as illustrated in Figure 3a. The common optimization solved by these techniques is:

\[ p = \arg \min_x ||f(x) - f(t)||^2 + \beta ||x - b||^2 \]

where \( f \) maps inputs to features, \( t \) is the target, \( b \) is a base example that is in a different class than \( t \), \( \beta \) balances the two objectives, and \( p \) is the generated poison example.

**End-to-end Training.** In end-to-end training, all layers of the model are trainable where any feature space that is not the input space may change after model training. Therefore, CPA’s optimization may not be effective because any distance on the feature space corresponding to each layer can change arbitrarily. Figure 3b illustrates this point where the poison example \( p \) can still be close to the base example \( b \) on a feature space that is not the input space even after CPA’s optimization. To clearly demonstrate this point, we run CPA on the AdultCensus dataset using a multilayer perceptron (see more experimental settings in Section 4) and generate a poison example \( p \). We then observe how the \( L_2 \) distances between \( p \) and \( t \) and between \( p \) and \( b \) change. As the model trains in a transfer learning scenario (Dotted), the feature distance from \( p \) to \( t \) decreases on three different layers in the model while the input distance from \( p \) to \( b \) remains small. However, when the model trains end-to-end (Solid), the feature distance from \( p \) to \( t \) increases rapidly, which means that the model no longer classifies \( p \) the same as \( t \).

### 3 REDACTOR

#### 3.1 Overview

We design an optimization problem of generating targeted disinformation for end-to-end training. We describe our objectives and introduce the overall process of Redactor. In end-to-end training, we can only utilize the input space and need to generate a disinformation that is as close as possible to the target example, but likely to be labeled as a different class from the target. Suppose that a human labeler has a mental decision boundary for labeling. In order to satisfy both conditions, the disinformation must be the closest point on the other side based on this decision boundary. Since we do not have control of the labeling and thus do not know the decision boundary, we propose to use surrogate models as a proxy for human labeling. When combining these models, our objective is not necessarily maximizing the overall accuracy, but instead looking for a conservative decision boundary that can confidently tell whether an example will be labeled differently than the target. An additional challenge is to make the disinformation as realistic as possible. For now, we assume there is a set \( C_{real} \subseteq \mathbb{R}^D \) that conceptually contains all possible realistic candidates where \( D \) is the number of features. In Section 3.4, we propose techniques for generating realistic examples.

We now formulate our optimization problem as follows:

\[
\min \sum_{j=1}^{N_d} ||d_j - t||^2 \\
\text{s.t. } \max_{c \notin t} M_c(\phi, d_j) \geq \alpha
\]

Figure 4: We run CPA [53] on the AdultCensus dataset and observe the relative \( L_2 \) distances from the initial points depicted in Figure 3. We measure distances from the poison example \( p \) to target \( t \) on possible feature spaces (different layers of a neural network). The transfer learning (TL) and end-to-end training (ET) scenario results are shown as dotted lines and solid lines, respectively. The TL results show how CPA is effective in reducing the distances from \( p \) to \( t \), while the ET results show how it fails to do so.
When choosing surrogate models, it is useful to have a variety of models that can complement each other in terms of performance. Similar strategies are used in data programming and ensemble learning. However, our goal is not necessarily improving the overall accuracy of the combined model, but ensuring a conservative probabilistic decision boundary. That is, there should be few false positives where a disinformation that is predicted to be on the other side of the target is actually labeled the same.

Another issue is that we may only have partial data for training surrogate models. Indeed, if we are protecting personal information on the Web, it is infeasible to train a model on the entire Web data. However, we argue that we only need data that is in the vicinity of the target and contains some examples in different classes as well. We only require that the probabilistic decision boundary approximates the decision making around the target. For example, entity resolution [13, 17] also has this issue for large data, and there are solutions for resolving subsets of data of interest [6] in order to perform entity resolution in query time. We assume the same setting, except that we are training models instead.

3.3 Probabilistic Decision Boundaries
We now explain how to generate a conservative probabilistic decision boundary for identifying examples that will very likely not be labeled the same as the target. We utilize multiple surrogate models and combine them into a single probabilistic model using data programming [39–41] techniques.

The data programming framework assumes that each labeling function can output a positive prediction (1), negative prediction (0), or abstained prediction (-1) if not confident enough. We implement a labeling function using a surrogate model \( \phi_i \) as follows where the \( \beta \) parameter is used to determine when to abstain:

\[
\lambda(\phi_i, x) = \begin{cases} 
1 & \phi_i(x) \geq 0.5 + \beta \\
0 & \phi_i(x) \leq 0.5 - \beta \\
-1 & \phi_i(x) - 0.5 < \beta 
\end{cases}
\]

We train a probabilistic generative model \( M \) with latent true labels \( Y \) using the label matrix \( \Lambda_{\phi, x} \) where \( \Lambda_{\phi, x} = \lambda(\phi_i, x) \) as follows:

\[
P_w(\Lambda_{\phi}, Y) = Z_w^{-1} \exp \left( \sum_{k=1}^{l} w^T \text{Corr}_k(\Lambda_{\phi}, y_k) \right)
\]

\[
\hat{w} = \arg \max_{w} \log \sum_{i} P_w(\Lambda_{\phi}, Y)
\]

\[
M(\phi, d) = P_w(Y | \Lambda_{\phi, d}).
\]

Here \( \text{Corr} \) indicates all possible correlations between labeling functions and latent \( Y \), \( Z_w^{-1} \) is the normalizing constant, and \( w \) has the weights of the generative model corresponding to each correlation.
We then use \( M \) as the probabilistic decision boundary. For each example \( d \), \( M \) returns a probability distribution of classes. Then \( d \) is considered to be in a different class than the target \( t \) if the class with the maximum probability is not \( t \)’s class, and the maximum probability is at least the tolerance threshold \( \alpha \).

3.4 Disinformation Generation
Given a target, we would like to find the closest possible points that would be labeled differently. Obviously we cannot use the target itself as it would not be labeled differently. Instead, we utilize the probabilistic decision boundary to find the closest point beyond the projected real decision boundary. We use watermarking [10, 27, 38, 44] techniques where a watermark of the target is added to the base example to generate disinformation using linear interpolations. While this approach works naturally for image data (i.e., the disinformation image is the same as the base image, but has a glimpse of the target image overlaid), structured data consists of numeric, discrete, and categorical features, so we need to perform watermarking differently. For numeric features, we can take linear interpolations. For discrete features that say require integer values, we use rounding to avoid outputting real numbers as a result of the interpolation. For categorical features, we choose the base’s value or target’s value, whichever is closer. More formally:

\[
\text{numeric} : d^{(i)} = \gamma r^{(i)} + (1 - \gamma) t^{(i)}
\]

\[
\text{discrete} : d^{(i)} = \text{round}(\gamma r^{(i)} + (1 - \gamma) b^{(i)})
\]

\[
\text{categorical} : d^{(i)} = \text{round}(\gamma) r^{(i)} + \text{round}(1 - \gamma) b^{(i)}
\]

where \( d \) is the disinformation example, \( t \) is the target, \( b \) is a base example, \( x^{(i)} \) is \( x \)’s attributes corresponding to the feature index set \( i \), \( \text{round}(x) = \lfloor x + 0.5 \rfloor \), and \( 0 \leq \gamma \leq 1 \).

In order to increase our chances of finding disinformation closer to the target, we can use GANs to generate more bases that are realistic and close to the decision boundary. Among possible GAN techniques for tabular data [5, 11, 35, 46, 51, 52], we extend the conditional tabular GAN (CTGAN) [51], which is the state-of-the-art method for generating realistic, but fake tabular data. CTGAN’s key techniques are using mode-specific normalization to learn complicated column distributions and training-by-sampling to overcome imbalanced training data.

Make Examples Realistic. CTGAN does not guarantee that all constraints requiring domain knowledge are satisfied. For example, in the AdultCensus dataset, the marital status “Wife” means that the person is female, but we need to perform separate checking instead of relying on CTGAN. Our solution is to avoid certain patterns
We evaluate Redactor and answer the following questions.

- Is a probabilistic decision boundary a good labeler proxy?
- How effective is Redactor’s disinformation in reducing model accuracy and confidence?
- How realistic is Redactor’s disinformation to humans?
- Can Redactor scale to large data by using partial data?

### 4 EXPERIMENTS

We evaluate Redactor and answer the following questions.

- Is a probabilistic decision boundary a good labeler proxy?
- How effective is Redactor’s disinformation in reducing model accuracy and confidence?
- How realistic is Redactor’s disinformation to humans?
- Can Redactor scale to large data by using partial data?

#### Algorithm 1: Pseudo code for generating disinformation.

```plaintext
Input: Target example \( t \), available data \( I \), trained surrogate models \( \phi \), trained generator model \( G \), number of disinformation examples \( N_d \), number of generated samples \( N_{gen} \), tolerance threshold \( \alpha \), abstain threshold \( \beta \)

Output: Disinformation examples \( R \)

// Generate candidate examples
1. \( C_{real} \leftarrow \emptyset \);

2. \( B \leftarrow \text{NearestExamples}(I, t, N_d) \) \( s.t. \ c_t \neq c_B \);

3. for \( i \) in 0...\( r \) do
   4. \( y \leftarrow i/r \);
   5. \( C_{real} \leftarrow \text{WaterMarking}(B, t, y) \);
   6. \( C_{GAN} \leftarrow \text{Generate}(N_{gen} \cdot n) \);
   7. \( C_{GAN} \leftarrow \text{FilterUnrealisticRecord}(C_{GAN}, I) \);
   8. \( C_{real} \leftarrow \text{NearestExamples}(C_{GAN}, t, N_{gen}) \);

// Decision boundary approximation
9. \( \phi_{topK} \leftarrow \text{SelectTopKmodels}(C_{real}, \phi, k) \);

10. for \( \phi_i \in \phi \) do
    11. \( \Phi_i \leftarrow \phi_i(I) \);
    12. \( \Lambda \leftarrow \text{LabelMatrixTransform}(\Phi, \beta) \);
    13. \( M \leftarrow \text{TrainLabelModel}(\Lambda, I) \);

// Generate disinformation examples
14. \( R \leftarrow \emptyset \);

15. for \( j \) in 1...\( N_d \) do
   16. \( d_j \leftarrow \min_{x \in C_{real}} (|x - t|^2 s.t. M_k(x) \geq \alpha) \);
   17. \( R.append(d_j) \);
   18. \( C_{real} \text{remove}(d_j) \);

return \( R \);
```

that are never seen in the original data. In our example, there are no examples where a Wife is a male, so we ignore all CTGAN-generated examples with this combination. Instead of identifying all these patterns manually, we automatically find them by looking at the original data and identifying feature pairs that have very low occurrences. In addition, we use clipping and quantization techniques to further make sure the feature values are valid.

#### 3.5 Putting Everything Together

Algorithm 1 shows the overall algorithm of Redactor. We first select random base examples that are preferably close to the target, but obviously have different labels according to our judgement (Step 2). We then generate candidate disinformation examples using watermarking and a CTGAN (Steps 3–8). We also construct the probabilistic decision boundary by combining good-performing surrogate models into a probabilistic model (Steps 9–13). Finally, we return the disinformation examples that are on the other side of the decision boundary from the target (Steps 14–19).

#### 4.1 Settings

**Datasets.** We use real tabular datasets for classification tasks. In particular, we use two datasets that contain people records when their information is leaked. We also use a large dataset to demonstrate the scalability of our techniques.

- AdultCensus [31]: Contains 45,222 people examples and is used to determine if one has a salary above a threshold of $50K per year. The features consists of age, education, gender, and occupation among others.
- COMPAS [4]: Contains 7,214 examples and is used to predict recidivism. The features include of name, gender, and ethnicity among others.
- Diabetes [47]: A large dataset that contains 100,000 records of diabetes patients in 130 US hospitals between 1999–2008.

**Target and Base Examples.** For each dataset, we choose 10 targets per dataset randomly. For each target, we choose \( k \) nearest examples with different labels as the base examples to generate \( k \) watermarked disinformation examples.

**Measures.** To evaluate a probabilistic decision boundary, we use precision, which is defined as the proportion of examples that are on the other side of the decision boundary from the target that actually have different ground truth labels. To evaluate a model’s performance, we measure the accuracy, which is the proportion of predictions that are correct, and use the confidence given by the model. For all measures, we always report percentages.

**Models.** We use three types of models: surrogate models for probabilistic decision boundaries, victim models to simulate inaccessible black-box models, and attack models that are used to perform inference attacks (only used in Section 4.4; also see Figure 1).

We first explain the surrogate models and naming conventions:

- Seven neural networks that have different combinations of the number of layers, the number of nodes per layer, and the activation function. We use the naming format \( s_{nn}A.X-Y \), which means a neural network that uses the activation function \( A \) (tanh, relu, log, and identity) and has \( X \) layers with \( Y \) nodes per layer.
- Two decision trees \( s_{tree} \) and two random forests \( s_{rf} \) using the Gini and Entropy purity measures.
- Four SVM models \( (s_{svm}) \) using the radial basis function (rbf), linear, polynomial, and sigmoid kernels.
- Three other models: gradient boosting \( (s_{gb}) \), AdaBoost \( (s_{ada}) \), and logistic regression \( (s_{logreg}) \).

Table 2 shows a total of 18 surrogate models and their respective train, test, and cross validation accuracies on the AdultCensus dataset. We use the cross validation accuracies to select the top-\( k \) performing surrogate models without knowledge of the test accuracies. Note that the cross validation and test accuracies are similar.

We then use the following groups of surrogate models: \( g \_all \) contains all the models, \( g \_top-k \) contains the top-performing surrogate models, \( g \_nn\_only \) contains the neural network models, \( g \_tree\_only \) contains the tree models, \( g \_svm\_only \) contains the SVM models, and \( g \_others \) contains the rest of the models.

For the victim models, we use a subset of Table 2 consisting 13 models (four neural networks, four trees and forests, two SVMs, and
### Table 2: 18 surrogate model architectures and their individual Train, Test, and Cross Validation (CV) accuracies on the AdultCensus dataset. The CV accuracies are needed to select the top-k performing models.

| Surrogate Model | Train Acc. | Test Acc. | CV Acc. |
|-----------------|------------|-----------|---------|
| tanh_5-2        | 86.46      | 85.07     | 84.24   |
| relu_5-2        | 86.57      | 85.24     | 84.92   |
| relu_50-25      | 90.33      | 82.44     | 82.67   |
| relu_200-100    | 95.55      | 81.56     | 81.64   |
| relu_25-10      | 87.93      | 84.22     | 83.64   |
| log_5-2         | 85.63      | 85.26     | 84.50   |
| identity_5-2    | 84.84      | 84.74     | 84.79   |
| s_nn            |            |           |         |
| dt_gini         | 85.28      | 85.56     | 84.73   |
| dt_entropy      | 85.29      | 85.38     | 84.84   |
| rf_gini         | 85.08      | 85.21     | 84.93   |
| rf_entropy      | 85.16      | 85.37     | 84.96   |
| s_tree          |            |           |         |
| rbf             | 85.83      | 84.92     | 84.53   |
| linear          | 84.78      | 85.03     | 84.63   |
| polynomial      | 85.13      | 83.16     | 82.79   |
| sigmoid         | 81.24      | 82.11     | 82.22   |
| s_svm           |            |           |         |
| s_gb            | 85.70      | 85.99     | 86.12   |
| s_ada           | 86.22      | 86.30     | 86.12   |
| s_logreg        | 84.90      | 84.86     | 84.76   |

Method. We compare Redactor with three baselines: (1) CPA is the convex polytope attack described in Section 2; (2) GAN only is Redactor using a CTGAN only; and (3) WM only is Redactor using watermarking only.

Other Settings. For all models, we set the learning rate to 1e-4 and the number of epochs to 1k. For CTGAN [51], we set the input random vector size to 100. We use PyTorch [36] and Scikit-learn [37], and all experiments are performed using Nvidia Titan RTX GPUs. We evaluate all models on separate test sets.

### 4.2 Decision Boundary as a Labeler Proxy

We evaluate the probabilistic decision boundary precision in Table 3, which shows the probabilistic decision boundary’s precision for different \( \alpha \) tolerance threshold values. As \( \alpha \) increases, the precision tends to increase except for model groups with fewer than five surrogate models. We observe that combining more surrogate models improves the precision as well, but only to a certain extent. Compared to taking a majority vote of surrogate models (MV), the precision of a probabilistic decision boundary is usually higher. In particular, using top-5 combined with \( \alpha = 0.95 \) results in the best precision. We thus use this setting in the remaining sections.

### Table 3: Precision for probabilistic decision boundaries with different \( \alpha \) tolerance thresholds (0.5–0.99) and taking a majority vote of the surrogate models (MV).

| Group        | 0.5   | 0.7   | 0.9   | 0.95  | 0.99  | MV  |
|--------------|-------|-------|-------|-------|-------|-----|
| g_all        | 83.68 | 83.86 | 84.13 | 84.35 | 84.38 | 84.42 |
| g_top-15     | 84.52 | 84.72 | 85.11 | 85.22 | 85.59 | 84.37 |
| g_top-12     | 84.87 | 84.87 | 85.11 | 85.25 | 85.79 | 83.81 |
| g_top-10     | 85.00 | 85.00 | 85.20 | 85.49 | 86.28 | 83.90 |
| g_top-7      | 85.33 | 85.82 | 86.28 | 86.96 | 88.28 | 83.66 |
| g_top-5      | 85.37 | 86.39 | 87.95 | 88.74 | 88.83 | 84.24 |
| g_top-3      | 85.30 | 87.18 | 82.45 | 82.45 | 82.74 | 75.39 |
| g_nn-only    | 81.74 | 81.79 | 82.08 | 82.32 | 82.66 | 84.43 |
| g_tree-only  | 85.44 | 86.12 | 87.86 | 88.18 | 79.34 | 84.35 |
| g_svm-only   | 82.96 | 82.96 | 82.96 | 82.96 | 86.20 | 84.83 |
| g_others     | 85.13 | 86.94 | 80.33 | 80.33 | 77.27 | 75.39 |

### 4.3 Disinformation Performance

We evaluate Redactor’s disinformation in terms of how it reduces a victim model’s accuracy and confidence on the AdultCensus and COMPAS datasets, respectively. We average the performances of the 13 victim models.

| Group        | Overall | Target | Target |
|--------------|---------|--------|--------|
|               | Test Acc. | Accuracy | Confidence |
| AdultCensus   | CPA -0.27±0.52 | -2.78±8.08 | -1.97±8.60 |
|               | GAN only -0.49±0.65 | -1.67±13.72 | -13.30±10.69 |
|               | WM only -1.43±1.40 | -28.89±12.78 | -21.35±15.00 |
|               | Redactor -1.99±1.73 | -37.22±13.20 | -26.23±14.44 |
| COMPAS        | CPA -0.26±0.80 | -0.56±5.39 | -2.24±3.10 |
|               | GAN only -0.14±0.72 | -5.56±10.96 | -2.30±3.31 |
|               | WM only -2.31±2.08 | -32.77±20.23 | -21.68±13.26 |
|               | Redactor -2.40±2.18 | -33.89±18.83 | -23.93±14.37 |
average target-disinformation $L_2$ distance of each cluster. As the disinformation is further away from a target, it becomes difficult to reduce the target's accuracy and confidence.

4.4 Defense Against Inference Attacks
Redactor can also defend against membership inference attacks (MIAs), which are the most popular inference attacks studied in the literature. The goal of an MIA is to train an attack model that predicts if a specific example was used to train a victim model based on its predictions and loss values. As explained in Section 4.1, we use 9 independent models in Table 2 with different hyperparameters for attacking the trained victim models. We use the AdultCensus dataset and select 10 target examples. Table 5 shows the MIA performances with and without 200 disinformation examples using the 9 attack models. For each scenario, we specify the attack model’s overall $F_1$ score and average target accuracy. We use the $F_1$ score just for this experiment to address the class imbalance of membership vs. non-membership. Each experiment is repeated seven times. The less accurate the attack model, the better the privacy of the target. As a result, the overall $F_1$ score of the attack model does not change much, but the target accuracy decreases significantly (by up to 26%) due to the disinformation. Furthermore, some target accuracies drop to around 50%, which means the classification is almost random.

4.5 Realistic Examples
We perform a comparison of our disinformation with real data to see how realistic it is. Recall in Section 3.4 that we filter out examples that contain feature pair patterns that do not occur in the original data. Table 6 shows a representative disinformation example (among many others) that was generated using our method along with the target and the target’s nearest examples. To see if the disinformation is realistic, we conduct a poll asking 11 human workers to correctly identify 5 disinformation and 5 real examples. As a result, the average accuracy is 53%, and the accuracies for identifying disinformation and real examples are 40% and 65%, respectively. We thus conclude that humans cannot easily distinguish our disinformation from real examples, and that identifying disinformation examples is harder than identifying real examples.

### Table 5: Using Redactor’s disinformation to defend against MIAs. For 10 target examples, a total of 200 disinformation examples are generated. For each attack model, we show how the disinformation changes its performances.

| Attack Model | Without Disinfo. | With Disinfo. | Target Acc. Change |
|--------------|-----------------|---------------|-------------------|
| a_tanh_5-2   | 58.96           | 59.41         | -12.86            |
| a_relu_5-2   | 61.18           | 61.33         | -11.43            |
| a_identity_5-2| 59.19           | 59.04         | -10.00            |
| a_dt_gini    | 52.02           | 51.04         | -26.66            |
| a_dt_entropy | 52.22           | 51.66         | -16.67            |
| a_rf_gini    | 52.20           | 52.03         | -13.33            |
| a_rf_entropy | 51.78           | 51.86         | -10.00            |
| a_logreg     | 53.85           | 52.74         | -7.14             |
| AVG          | 55.86           | 55.59         | -10.00            |

Figure 7: A comparison of average local surrogate model accuracies when training the models on the Diabetes dataset.

4.6 Scalability
If a dataset is too large, we can still run Redactor on partial data as explained in Section 3.2. We evaluate Redactor on the large Diabetes dataset by first selecting 10 random targets and then training the surrogate models on nearest neighbors of the targets where we maintain a balance of different classes. As long as the surrogate models are accurate, we know Redactor performs well. Figure 7 shows that, as the number of nearest neighbors increases, the average local accuracy on a separate test set of other nearest neighbors rapidly increases. In particular, using 2,000 nearest neighbors (2% of entire data) gives an average local accuracy comparable (within 1%) to the average accuracy of training models on the entire dataset.

5 RELATED WORK
Redactor is related to multiple disciplines, and we explain why it solves a novel problem. The objective of generating disinformation is related to data privacy and deletion, although the problem setting is different where there is a single target to protect, and there is no control over the model or data. The techniques of Redactor are closely related to data poisoning attacks, although we assume the more common setting of end-to-end training instead of a transfer learning setting. Finally, Redactor generates realistic disinformation primarily for structured data.

Data Privacy, Deletion, and Disinformation. Data privacy is a broad discipline of protecting one’s personal information within
data. The most popular approach is differential privacy [14–16] where random records are added to a database to lower the chance of information leakage. In comparison, our problem is a special case of data privacy where there is no control over the data, and the only way to improve one’s privacy is to add disinformation.

A related problem is data deletion where the goal is to make a model forget about certain data. Data deletion has been recently studied when using $k$-means clustering [20], non-iterative ML models [43], $L_2$-regularized linear regression [25, 28], neural networks (scrubbing [21, 22] and forgetting [8, 24, 50]). Most of these techniques assume that the data or model can be changed or updated at will. In comparison, we only assume that data can be added and that models may be trained with the new data at some point. We also do not assume label information, but do not control the end model either.

The concept of disinformation is not new and has been studied in different contexts. A work on data leakage detection [34] uses disinformation to determine whether any information has been leaked when data is distributed. A work on entity resolution [49] proposes optimization techniques for lowering the entity resolution accuracy while using a limited budget for generating disinformation. In comparison, Redactor focuses on obfuscating information in ML models for data privacy.

**Table 6: Comparison of disinformation with real data using the AdultCensus dataset.**

| Target | Disinformation | Nearest Examples to Target |
|--------|----------------|---------------------------|
| Age    | 38             | 43                        |
| Workclass | Private     | Private                 |
| Education | HS-grad       | HS-grad                  |
| Marital status | Never-married | Never-married            |
| Occupation | Machine-op-inspect | Machine-op-inspect      |
| Relationship | Not-in-family | Not-in-family            |
| Race    | White          | White                    |
| Gender  | Male           | Male                     |
| Capital gain | 0             | 7676                     |
| Hrs/week | 40            | 40                       |
| Country | US            | US                       |
| Income  | $\leq$50K     | US $>$50K                |

**Data Poisoning.** Targeted poisoning attacks [30, 44, 48, 53] have the goal of flipping the prediction of specific targets to a certain class. Clean-label attacks [44, 48] have been proposed for neural networks to alter the model’s behavior on a specific test instance by poisoning the training set without having any control over the labeling. Convex Polytope Attack (CPA) [53] is another type of clean-label targeted poisoning and has been proposed to generate examples that are similar to the base examples, but have similar predictions as a target. CPA is not involved in the model training, but generates poisoned examples that are presumably added to the training set. The goal is to generate examples close to the target in the feature space while close to a base in the input space. CPA performs well for image datasets where transfer learning is used. Both of these techniques primarily target a transfer learning setting whereas Redactor is designed for end-to-end training.

Exploratory attacks are used to extract information from models, and various defenses have been proposed. The dominant attack most related to our work is the membership inference attack, and many defenses [29, 32, 42] have been proposed. However, most works assume access to the victim’s model. For example, MemGuard [29] is a state-of-the-art defense that adds noise to the model’s output to drop the attack model’s performance. Other techniques include adding a regularizer to the model’s loss function [32] and applying dropout or model stacking techniques [42]. However, such model modifications are not possible in our setting where we assume no access to the model.

**Tabular Data Generation.** Generating fake tabular data is becoming a major area in GAN research [9, 35, 51, 52]. CTGAN [51] generates realistic data, but obviously does not necessarily satisfy all necessary constraints as they require domain knowledge. FakeTables [9] focuses on satisfying functional dependencies when normalizing tables. LowProFool [5] generates adversarial examples that are imperceptible by only modifying relatively unnoticeable features. Redactor can utilize any of these techniques for generating realistic base examples.

**6 CONCLUSION**

We proposed effective targeted disinformation methods for black box models on structured data where there is no access to the labeling or model training. We explained why an end-to-end training setting is important and that existing poisoning attacks that rely on a transferable learning setting do not perform well. We then presented Redactor, which is designed for end-to-end training where it generates a conservative probabilistic decision boundary to emulate labeling and then generates realistic disinformation examples that reduce the target’s accuracy and confidence the most. Our experiments showed that Redactor generates disinformation more effectively than other poisoning attacks, defends against membership inference attacks, and generates realistic disinformation.
