Redactor: A Data-Centric and Individualized Defense Against Inference Attacks

Geon Heo, Steven Euijong Whang

KAIST
{geon.heo, 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 trainings by other parties either. In this setting, we study the problem of *targeted disinformation generation* where the goal is to dilute the data and thus make a model safer and more robust against inference attacks on a specific target (e.g., a person's profile) by only inserting new data. Our method finds the closest points to the target in the input space that will be labeled as a different class. Since we cannot control the labeling process, we instead conservatively estimate the labels probabilistically by combining decision boundaries of multiple classifiers using data programming techniques. Our experiments show that a probabilistic decision boundary can be a good proxy for labelers, and that our approach is effective in defending against inference attacks and can scale to large data.

Introduction

Information leakage is becoming a serious problem as personal data is being used to train machine learning (ML) models. Personal data can be leaked through AI chatbots (McCurry 2021) and the Web (Hill and Krolik 2019) among others. Furthermore, there are various privacy threats on ML models including inference attacks (Shokri et al. 2017; Hayes et al. 2019; Choo et al. 2020) and reconstruction attacks (Fredrikson, Jha, and Ristenpart 2015). Defending against such leakage is critical for safe and robust AI.

In many cases, it is impossible to delete one's information that is published on the Web or uploaded on a third-party platform. 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. Therefore, conventional privacy techniques or defenses that require ownership of the data or model cannot be used here.
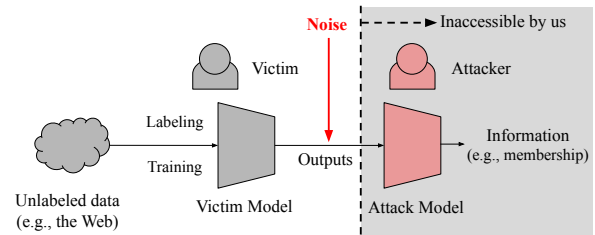
(a) Conventional Inference Attack Defense
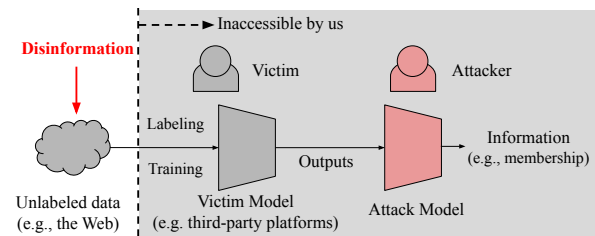
(b) Redactor

Figure 1: (a) Existing inference attack defenses (e.g., add noise to model's output) are not feasible when the victim models are owned by third parties. (b) Instead, we assume the realistic setting where we can only add disinformation to the unlabeled training data, which is presumably labeled and used for model training by unknown model owners.

The only solution is to take a data-centric approach and add new data that "dilutes" an individual's personal information, which we refer to as *disinformation*. An analogy is blacking out or redacting text where the reader knows there is some information, but cannot read it. We thus define the problem of *targeted disinformation generation* where the goal is to generate disinformation that indirectly makes a victim model less likely to leak personal information to an attack model without any access to the victim model (Figure 1). We assume the disinformation will be eventually picked up automatically by crawlers for model training, which is a common assumption in the AI Security literature (Shafahi et al. 2018; Suciu et al. 2018; Chen et al. 2019). From an ethical perspective, our disinformation is intended to protect one's information from inference attacks.

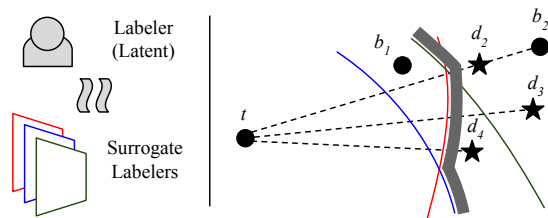Our solution is motivated by clean-label targeted poison-

Figure 2: A labeler is approximated as a probabilistic model that combines surrogate labelers. The decision boundaries of surrogate labelers are shown as thin lines while the probabilistic decision boundary (PDB) is the thick gray line. Using the base example b_2, Redactor can generate the disinformation example d_2 that is close to t, but still labeled differently. In addition, Redactor can generate other realistic points d_3 and d_4 using a generative model where d_4 happens to be even closer to t.

ing methods (Suciu et al. 2018; Shafahi et al. 2018; Zhu et al. 2019; Kermani et al. 2015) that degrade the model performance only on a target example. We would like to utilize such techniques to change the output of the unknown models (e.g., third-party platforms) and protect target examples from indirect privacy attacks. However, many of these poisoning techniques implicitly rely on a transfer learning (Pan and Yang 2010) setup where a pre-trained model is available and exploit the fact that the feature space is fixed for optimizing the poisoning. While transfer learning benefits certain applications (e.g., NLP or vision tasks), it is not always available, especially for structured data where there is no efficient and generally-accepted practice (Borisov et al. 2021). However, structured data is key to our problem as most personal information is stored in this format (e.g., name, gender, age, race, address, and more). In order to primarily support structured data, we thus need to assume *end-to-end training* where we cannot count on a fixed feature space. Although existing techniques have also been extended for end-to-end training, we show their performances are not sufficient.

Since we cannot rely on a fixed feature space, we instead 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, one of our key contributions is a novel adaptation of data programming (Ratner et al. 2020, 2017a,b) to conservatively estimate human behavior using a *probabilistic decision boundary* (PDB) produced by combining multiple possible classifiers. In our setting, we make the generative model produce the probability of an example having a class that is different than the target's class. By limiting this probability to be above a tolerance threshold, we now have a conservative decision boundary. This approach is agnostic to the victim model. We call our system Redactor, and Figure 2 illustrates our overall approach.

Our contributions: (i) We define the *targeted disinformation generation* problem. (ii) We suggest a novel data-centric and model-agnostic defense using data programming and generative models without any access to the training process

of victim models. (iii) We empirically demonstrate that our solution is effective in defending against membership inference attacks and scales to large data.

Background

Membership Inference Attack (MIA)

Among various types of adversarial attacks, exploratory attacks are used to extract information from models. The dominant attack most related to our work is the membership inference attack (MIA). 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 confidence scores and loss values. Formally, an attacker trains an attack model A satisfying $A : s \rightarrow \{0, 1\}$ where the input s is the confidence score or loss value of the victim model V for an example x, and 1 means that x is a member of the training set of V.

Many defenses (Jia et al. 2019; Li, Li, and Ribeiro 2021; Salem et al. 2019) have been proposed against MIAs, but most of them assume that accessing the victim model V is possible. For example, MemGuard (Jia et al. 2019) 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 (Li, Li, and Ribeiro 2021) and applying dropout or model stacking techniques (Salem et al. 2019).

Such model modifications are not possible in our setting where we assume no access to the model. We thus design a new approach using a targeted poisoning objective to indirectly change the victim model's performance on the target (confidence score and loss value).

Targeted Poisoning

Targeted poisoning attacks have the goal of flipping the predictions on specific targets to a certain class. A naïve approach is to add examples that are identical to the target, but with different labels. Unfortunately, such an approach would only work if one has 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.

The state-of-the-art targeted poisoning attacks include Convex Polytope Attack (CPA) (Zhu et al. 2019) and its predecessors (Chen et al. 2017; Suciu et al. 2018; Shafahi et al. 2018), 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. These techniques are not involved in the model training itself, but generate poisons 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. To find an optimal poison satisfying such conditions, CPA utilizes a fixed feature extractor, which is effective when the victim uses transfer learning (see Figure 3a).

In end-to-end training, however, all the layers of the model are trainable where any feature space that is not the input space may change after model training with the poison. Therefore, CPA's optimization may not be effective because any distance on the feature space corresponding to each

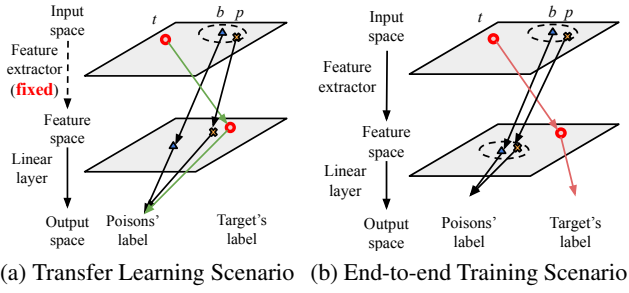(a) Transfer Learning Scenario   (b) End-to-end Training Scenario

Figure 3: In a transfer learning scenario (a), the feature space is fixed, making it possible to optimize on both the input and feature spaces. In an end-to-end scenario (b), however, the feature space may change after the model trains, so optimizing on the feature space may not be effective.

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. Empirical results for this analysis can be found in our technical report (2022). Although (Zhu et al. 2019) suggests the extension of applying CPA on every layer of the network for end-to-end training, it is inefficient and does not fundamentally solve the problem. We thus need a completely different solution that does not utilize the feature space for optimization.

Methodology: Redactor

We design an optimization problem of generating targeted disinformation for end-to-end training based on the targeted poisoning objective. 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 as we define below:

$$\min_{\{d_j\}} \sum_{j=1}^{N_d} dist(d_j, t)$$

$$\text{s.t. } HumanLabel(d_j) \neq c_t$$

$$d_j \in C_{real}, \ \forall j \in [1 \dots N_d]$$

(1)

where $t \in \mathbb{R}^D$ is the target example, $d_j \in \mathbb{R}^D$ is the jth disinformation among a budget of N_d disinformations, c_t is t's class, and $C_{real} \subseteq \mathbb{R}^D$ is a set that conceptually contains all possible realistic candidates where D is the number of features. However, since we do not have control of the labeling and thus do not know the decision boundary, we propose to use surrogate classifier models as a proxy for human labeling, which we call *surrogate labelers*. This approach is inspired by ensemble techniques commonly used for forging black-box attacks (Zhu et al. 2019; Liu et al. 2017). We do not assume that the surrogate labelers are highly accurate.
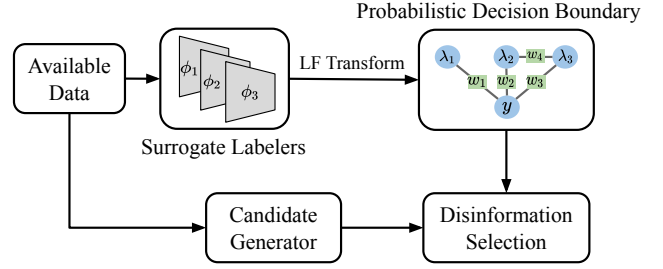

Figure 4: Redactor runs in four stages: surrogate labeler training, Probabilistic Decision Boundary (PDB) creation, candidate generation, and disinformation selection.

However, when combining these models, we assume that we can find a *conservative decision boundary* that can confidently tell whether an example will be labeled differently than the target. Based on Equation 1, we now formulate our optimization problem as follows:

$$\min_{\{d_j\}} \sum_{j=1}^{N_d} ||d_j - t||^2$$

$$\text{s.t. } \arg\max_c M_c(\phi, d_j) \neq c_t$$

$$\max_{c \neq c_t} M_c(\phi, d_j) \geq \alpha$$

$$d_j \in C_{cand}, \ \forall j \in [1 \dots N_d]$$

(2)

where $M_c(\phi, x)$ is the probabilistic generative model that combines surrogate labelers ϕ and returns the probability of an example x being in class c, C_{cand} is a realistic candidate set that we generate, and α is the tolerance threshold for M_c. We use common pre-processings where numeric features are normalized, and categorical features are converted to have numerical values using one-hot encoding.

Redactor generates disinformation in four stages (Figure 4): training surrogate labelers on the available data, creating a PDB, generating realistic candidates, and selecting the examples that will be used as disinformation. In the next sections, we cover each component in more detail. The overall algorithm is in our technical report (2022).

Training Surrogate Labelers

When choosing surrogate labelers, it is useful to have a variety of models that can complement each other in terms of performance as they are not necessarily highly accurate. Similar strategies are used in data programming and ensemble learning (Breiman 1996; Freund, Schapire et al. 1996). However, our goal is not necessarily improving the overall accuracy of the combined model, but ensuring a conservative PDB. 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 labelers because the data is too large or unavailable. 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 PDB approximates the decision making around the target. In our experiments, we show how Redactor can scale to large data by properly selecting partial data.

Probabilistic Decision Boundary (PDB)

We now explain how to generate a conservative PDB for identifying examples that will very likely not be labeled the same as the target. We utilize multiple surrogate labelers and combine them into a single probabilistic model using data programming (Ratner et al. 2020, 2017a,b) techniques, which combines multiple labeling functions into a label model that produces probabilistic labels.

The data programming framework assumes that each labeling function (LF) can output one of the classes as a prediction or an abstained prediction (-1) if not confident enough. The abstained prediction is necessary for making the LFs as reliable as possible. Thus, we transform each surrogate labeler ϕ_i as follows:

$$\lambda(\phi_i,\ x) = \begin{cases} -1(Abstain) & \max_c \phi_i^{(c)}(x) \leq \beta/C \\ \arg\max_c \phi_i^{(c)}(x) & \text{otherwise} \end{cases}$$

where β is used to determine when to abstain, $c \in [1 \ldots C]$ is a class, and ϕ_i outputs a C-dimensional probability vector.

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

$$P_w(\Lambda_\phi,\ Y) = Z_w^{-1} \exp\left(\sum_{k=1}^{l} w^T\ Corr_k(\Lambda_\phi,\ y_k)\right)$$

$$\hat{w} = \arg\max_w \log\sum_Y P_w(\Lambda_\phi,\ Y)$$

$$M(\phi,\ d) = P_{\hat{w}}(Y \mid \Lambda_{\phi,d}).$$

Here $Corr$ values are binary values indicating all possible correlations between LFs and the latent Y, Z_w^{-1} is the constant for normalization, and w has the weights of the generative model corresponding to each correlation.

We then use M with ϕ as the PDB. For each example d, M returns probability values for each class. 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 α.

Candidate Generation & Disinformation Selection

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 PDB to find the closest point beyond the projected real decision boundary. We use watermarking (Quiring, Arp, and Rieck 2018; Chen et al. 2017; Hitaj and Mancini 2018; Shafahi et al. 2018) 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:

Numeric : $d^{(i)} = \gamma t^{(i)} + (1 - \gamma)b^{(i)}$

Discrete : $d^{(i)} = round(\gamma t^{(i)} + (1 - \gamma)b^{(i)})$

Categorical : $d^{(i)} = round(\gamma)t^{(i)} + round(1 - \gamma)b^{(i)}$

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, $round(x) = \lfloor x + 0.5 \rfloor$, 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 (Ballet et al. 2019; Choi et al. 2017; Srivastava et al. 2017; Park et al. 2018; Xu and Veeramachaneni 2018; Xu et al. 2019), we extend the conditional tabular GAN (CTGAN) (Xu et al. 2019), which is the state-of-the-art method for generating realistic 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.

Realistic Examples   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 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. This checking can be performed efficiently by identifying frequent feature pairs in the original data and rejecting any feature pair that does not appear in this list. In addition, we use clipping and quantization techniques to further make sure the feature values are valid.

Experiments

Datasets   We use four real tabular datasets for binary and multi-class classification tasks. All the datasets contain people records whose information can be leaked. The last Diabetes dataset is large and thus used to demonstrate the scalability of our techniques.

- AdultCensus (Kohavi 1997): Contains 45,222 people examples and is used to determine if one has a salary of $\geq$\$50K per year.
- COMPAS (Angwin et al. 2016): Contains 7,214 examples and is used to predict criminal recidivism rates.
- Epileptic Seizure Recognition (ESR) (Andrzejak et al. 2001): Contains 11,500 electroencephalographic (EEG) recording data and is used to classify five types of brain states including epileptic seizure.
- Diabetes (Strack et al. 2014): Contains 100,000 diabetes patient records in 130 US hospitals between 1999–2008.

	Surrogate Labelers
s_nn	*relu_5-2, relu_50-25, relu_200-100, relu_25-10* *tanh_5-2, log_5-2, identity_5-2*
s_tree	*dt_gini, dt_entropy, rf_gini, rf_entropy*
s_svm	*rbf, linear, polynomial, sigmoid*
others	*s_gb, s_ada, s_logreg*

Table 1: 18 surrogate labeler architectures.

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 PDB, we use *precision*, which is defined as the portion of examples that are on the other side of the decision boundary from the target and have different ground truth labels. To evaluate a model's performance, we measure the *accuracy*, which is the portion 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 labelers* for PDBs, *victim models* to simulate inaccessible black-box models, and *attack models* that are used to perform MIAs.

We use 18 surrogate labelers explained below and summarized in Table 1. Although we could use more complex models, they would overfit on our datasets.

- 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 convention *s_nn_A_X-Y*, which indicates 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*).

For the victim models, we use a subset of Table 1 consisting of 13 models (four neural networks, four trees and forests, two SVMs, and three others), but with different numbers of layers and optimizers to clearly distinguish them from the surrogate labelers. For the attack models, we select nine of the smallest models having the fewest layers, depth, or number of tree estimators from Table 1. We choose small models because attack models train on a victim model's output and loss and need to be small to perform well. We use the same naming conventions as Table 1 except that the model names start with "a_" instead of "s_" as shown in Table 4.

Methods We compare Redactor with three baselines: (1) *CPA* is the convex polytope attack extended to end-to-end training; (2) *GAN only* is Redactor using a CTGAN only; and (3) *WM only* is Redactor using watermarking only.

Group	0.5	0.7	0.9	0.95	0.99	MV
all	83.68	83.86	84.13	84.35	84.38	84.42
top-15	84.52	84.72	85.11	85.22	85.59	84.37
top-10	85.00	85.00	85.20	85.49	**86.28**	83.90
top-5	85.37	86.39	**87.95**	**88.74**	78.92	84.24
top-3	85.30	**87.18**	82.45	82.45	78.54	75.39
nn-only	81.74	81.79	82.08	82.32	82.66	84.43
tree-only	**85.44**	86.12	87.86	88.18	79.34	84.35
svm-only	82.96	82.96	82.96	82.96	64.20	**84.83**
others	85.13	86.94	80.33	80.33	77.27	75.39

Table 2: Precision for PDBs with different α tolerance thresholds (0.5–0.99) and taking a majority vote of the surrogate labelers (MV).

Other Settings We set the abstain threshold β to 0.1. For all models, we set the learning rate to 1e-4 and the number of epochs to 1K. For CTGAN, we set the input random vector size to 100. We use PyTorch and Scikit-learn, and all experiments are performed using Nvidia Titan RTX GPUs. We evaluate all models on separate test sets.

Decision Boundary as a Labeler Proxy

We evaluate the PDB precision in Table 2. We use cross validation accuracies to select the top-k performing surrogate labelers without knowledge of the test accuracies (more details are in our technical report (2022)). We then use the following groups of models as surrogate labelers: *all* contains all the models, *top-k* contains the top-performing surrogate labelers, *nn_only* contains the neural network models, *tree_only* contains the tree models, *svm_only* contains the SVM models, and *others* contains the rest of the models. The table shows the PDB's precision for different α tolerance thresholds. For model groups with many surrogate labelers, the precision tends to increase for larger α values.

We observe that employing five or more surrogate labelers leads to good performance. Compared to taking a majority vote of surrogate labelers (MV), the precision of a PDB is usually higher. In particular, using *top-5* combined with $\alpha = 0.95$ results in the best precision, so we use this setup in the remaining sections.

Disinformation Performance

We evaluate Redactor's disinformation in terms of how it changes a victim model's accuracy and confidence on the AdultCensus, COMPAS, and ESR in Table 3. (Evaluations on image data give similar results and are shown in our technical report (2022).) We train the 13 victim models for each dataset. We then generate disinformation for the targets (500 examples for AdultCensus, 50 for COMPAS, and 100 for ESR) and re-train the victim models on the dataset plus disinformation. As a result, Redactor reduces the performances more than the other baselines (especially CPA) without reducing Test Acc. significantly.

Using the same victim model setting, we also analyze how the disinformation budget and the distance between the tar-

		Overall Test Acc. Change	Target Acc. Change	Target Conf. Change
Adult Census	*CPA*	-0.27±0.52	-2.78±8.08	-1.97±8.60
	GAN only	-0.49±0.65	-16.67±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**
ESR	*CPA*	-0.43±4.27	-7.14±16.04	-2.59±5.75
	GAN only	-0.65±1.58	-8.57±15.74	-1.57±10.37
	WM only	-0.07±1.13	-34.29±12.72	-18.42±17.16
	Redactor	-0.11±0.89	**-35.71±13.97**	**-18.28±17.25**

Table 3: Average performance change of victim models on targets when generating disinformation examples on the AdultCensus, COMPAS, and ESR datasets. The number of inserted examples is about 1% of the entire dataset size.
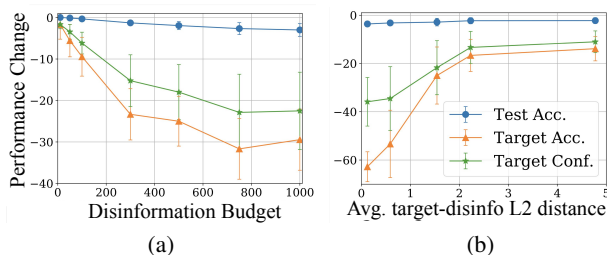

Figure 5: (a) As the number of disinformation examples that we can create increases, the target accuracy and confidence decrease significantly while the overall test accuracy decreases only by 2%. (b) As the distance to the target increases, we observe increasing trends as opposed to (a).

get and disinformation impacts the disinformation performance. We first select 10 random target examples and vary the number of disinformation examples generated. Figure 5a shows how the average target accuracy and confidence of the 13 victim models decrease further as more disinformation budget is allowed, but eventually plateaus. Next, we select 50 random target examples and generate disinformation. Then we cluster the targets by their average L_2 distances to their disinformation examples. We then plot each cluster in Figure 5b, which shows the average target accuracy and confidence degradation of the 13 models against the 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.

Defense Against Inference Attacks

We evaluate Redactor against MIAs (Shokri et al. 2017). We use nine models in Table 1 with different hyperparameters

Attack Model	Without Disinfo.		With Disinfo.		Target Acc. Change
	Overall F1 score	Target Acc.	Overall F1 score	Target Acc.	
a_tanh_5-2	58.96	78.57	59.41	65.71	-12.86
a_relu_5-2	61.18	82.86	61.33	71.43	-11.43
a_identity_5-2	59.19	75.71	59.04	65.71	-10.00
a_dt_gini	52.02	73.33	51.04	46.67	-26.66
a_dt_entropy	52.22	60.00	51.66	43.33	-16.67
a_rf_gini	52.20	65.00	52.03	51.67	-13.33
a_rf_entropy	51.78	55.71	51.86	45.71	-10.00
a_ada	53.85	61.43	52.74	54.29	-7.14
a_logreg	61.30	80.00	61.20	70.00	-10.00
Average	55.86	**70.29**	55.59	**57.17**	-13.12

Table 4: Using Redactor's disinformation to defend against MIAs using attack models. For 10 target examples, a total of 200 disinformation examples are generated. For each model, we show how the disinformation changes its performances.

for attacking the trained victim models. We use the Adult-Census dataset and select 10 random target examples. Table 4 shows the MIA performances with and without 200 disinformation examples using the nine attack models. For each scenario, we specify the attack model's overall F_1 score and average *target inference accuracy*, which is the fraction of target examples the attack model correctly predicts membership. We use the F_1 score just for this experiment to address the class imbalance of membership versus 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 26.66%) due to the disinformation. Furthermore some target accuracies drop to around 50%, which means the classification is almost random. Evaluations on other MIAs (Yeom et al. 2018; Salem et al. 2019) give similar results and are shown in our technical report (2022).

Realistic Examples

We perform a comparison of our disinformation with real data to see how realistic it is. Table 5 shows a representative disinformation example (among many others) that was generated using our method along with the target and a few nearest neighbors. To see if the disinformation is realistic, we conduct a poll asking 33 human experts to correctly identify five disinformation and five real examples. As a result, the average accuracy is 56.9%, and the accuracies for identifying disinformation and real examples are 46.1% and 67.8%, respectively. We thus conclude that humans cannot easily distinguish our disinformation from real data, and that identifying disinformation is harder than identifying real data.

Scalability

If the dataset is too large or not fully accessible, Redactor can still run on partial data. We evaluate Redactor on the Di-

	Age	Education	Marital status	Occupation	Relationship	Race	Gender	Capital gain	Hrs/week	Country	Income
T	38	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	40	US	≤50K
D	43	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	7676	40	US	>50K
NN	52	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	45	US	>50K
	36	HS-grad	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	7298	40	US	>50K

Table 5: Comparison of disinformation D of target T with T's nearest neighbors NN using the AdultCensus dataset.

	Dist. Known			Dist. Unknown		
n	Time (s)	Local Acc.	Δ Acc.	Time (s)	Local Acc.	Δ Acc.
1k	59.72	58.82	-30.00	47.35	66.64	-31.67
3k	100.52	68.69	-30.00	79.85	69.30	-30.00
5k	125.76	70.40	-31.67	109.28	69.84	-27.77
7k	162.78	72.30	-26.67	122.68	70.42	-20.37
All	2,043.1	70.67	-36.67	2,043.1	70.67	-36.67

Table 6: Evaluation of the two partial data strategies on the Diabetes dataset. For different data sizes (n), we show the average runtime for disinformation generation per target in seconds (Time), average local accuracy (Local Acc.), and average target accuracy change (Δ Acc.).

abetes dataset by selecting 10 random targets, training the 18 surrogate labelers on nearest neighbors of the targets using Euclidean distance, and generating 200 disinformation examples per target. We use two strategies for selecting the partial data: (1) *Dist. Known*: We assume the entire class distribution is known and collect n nearest neighbors of targets following this distribution (i.e., we effectively take a uniform sample of the entire data that is closest to the target) and (2) *Dist. Unknown*: We assume the distribution is unknown and collect nearest neighbors of the target until we have at least n examples per class.

In Table 6, we compare the following for different n values: (1) the average runtime for disinformation generation per target, (2) the average local accuracy, which is the average accuracy of surrogate labelers on the 10K nearest neighbors of each target, and (3) the average target accuracy change. As a result, when n is at least 3,000 (3% of the entire data), the runtime improves by >20x, while the average local accuracy and target accuracy change are comparable to the results using the entire data (All). In addition, utilizing the data distribution sometimes gives worse results than not due to the adjustment of class ratios of nearest neighbors to follow the entire distribution. Hence, using partial data without knowing the entire data distribution can be sufficient for effective disinformation.

Related Work

Data Privacy, Data Deletion, and Disinformation Data privacy is a broad discipline of protecting one's personal information within data. The most popular approach is differ-

ential privacy (Dwork et al. 2006; Dwork 2011; Dwork and Roth 2014) where random records are added to a database to lower the chance of information leakage. In comparison, we solve a subproblem of data privacy in ML where there is no control over the training data, and the only way to improve one's privacy is to add disinformation.

Another related problem is data deletion where the goal is to make a model forget about certain data (Ginart et al. 2019; Guo et al. 2020; Golatkar, Achille, and Soatto 2020a,b; Graves, Nagisetty, and Ganesh 2021)). Most of these techniques assume that the data or model can be changed 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.

Finally, disinformation has been studied in different contexts including data leakage detection (Papadimitriou and Garcia-Molina 2011) and entity resolution (Whang and Garcia-Molina 2013). In comparison, Redactor focuses on obfuscating information in ML models for data privacy.

Targeted Poisoning Targeted poisoning attacks (Suciu et al. 2018; Shafahi et al. 2018; Zhu et al. 2019; Kermani et al. 2015) have the goal of flipping the predictions of specific targets to certain classes. Clean-label attacks (Suciu et al. 2018; Shafahi et al. 2018; Zhu et al. 2019) 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) (Zhu et al. 2019) covers various structures of neural networks, which is different from other techniques. All these techniques rely on a fixed feature space for optimization whereas Redactor does not assume this.

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 targeted poisoning attacks that implicitly 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 targeted poisoning attacks, defends against MIAs, generates realistic disinformation, and scales to large data.

Acknowledgments

References

Andrzejak, R. G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; and Elger, C. E. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6): 061907.

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. Machine bias: There's software used across the country to predict future criminals. And its biased against blacks. ProPublica.

Ballet, V.; Renard, X.; Aigrain, J.; Laugel, T.; Frossard, P.; and Detyniecki, M. 2019. Imperceptible Adversarial Attacks on Tabular Data. *CoRR*, abs/1911.03274.

Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; and Kasneci, G. 2021. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*.

Breiman, L. 1996. Bagging predictors. *Machine learning*, 24(2): 123–140.

Chen, H.; Jajodia, S.; Liu, J.; Park, N.; Sokolov, V.; and Subrahmanian, V. S. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *IJCAI*, 2074–2080.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR*, abs/1712.05526.

Choi, E.; Biswal, S.; Malin, B. A.; Duke, J.; Stewart, W. F.; and Sun, J. 2017. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *MLHC*, volume 68, 286–305. PMLR.

Choo, C. A. C.; Tramer, F.; Carlini, N.; and Papernot, N. 2020. Label-Only Membership Inference Attacks. *arXiv preprint arXiv:2007.14321*.

Dwork, C. 2011. A firm foundation for private data analysis. *Commun. ACM*, 54(1): 86–95.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. D. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, volume 3876, 265–284. Springer.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4): 211–407.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 1322–1333.

Freund, Y.; Schapire, R. E.; et al. 1996. Experiments with a new boosting algorithm. In *icml*, volume 96, 148–156. Citeseer.

Ginart, A.; Guan, M. Y.; Valiant, G.; and Zou, J. 2019. Making AI Forget You: Data Deletion in Machine Learning. In *NeurIPS*, 3513–3526.

Golatkar, A.; Achille, A.; and Soatto, S. 2020a. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *CVPR*, 9301–9309.

Golatkar, A.; Achille, A.; and Soatto, S. 2020b. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In *ECCV*, 383–398.

Graves, L.; Nagisetty, V.; and Ganesh, V. 2021. Amnesiac Machine Learning. In *AAAI*, 11516–11524.

Guo, C.; Goldstein, T.; Hannun, A. Y.; and van der Maaten, L. 2020. Certified Data Removal from Machine Learning Models. In *ICML*, 3832–3842.

Hayes, J.; Melis, L.; Danezis, G.; and De Cristofaro, E. 2019. LOGAN: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1): 133–152.

Heo, G.; and Whang, S. E. 2022. Redactor: A Datacentric and Individualized Defense Against Inference Attacks. Technical report. https://arxiv.org/abs/2202.02902.

Hill, K.; and Krolik, A. 2019. How Photos of Your Kids Are Powering Surveillance Technology. https://www.nytimes.com/interactive/2019/10/11/technology/flickr-facial-recognition.html. Accessed 2022-08-03.

Hitaj, D.; and Mancini, L. V. 2018. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *arXiv preprint arXiv:1809.00615*.

Jia, J.; Salem, A.; Backes, M.; Zhang, Y.; and Gong, N. Z. 2019. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *CCS*, 259–274. ACM.

Kermani, M. M.; Sur-Kolay, S.; Raghunathan, A.; and Jha, N. K. 2015. Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare. *IEEE J. Biomed. Health Informatics*, 19(6): 1893–1905.

Kohavi, R. 1997. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. *KDD*.

Li, J.; Li, N.; and Ribeiro, B. 2021. Membership Inference Attacks and Defenses in Classification Models. In *CODASPY*, 5–16. ACM.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *ICLR*.

McCurry, J. 2021. South Korean AI chatbot pulled from Facebook after hate speech towards minorities. https://www.theguardian.com/world/2021/jan/14/time-to-properly-socialise-hate-speech-ai-chatbot-pulled-from-facebook. Accessed 2022-08-03.

Pan, S. J.; and Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.*, 22(10): 1345–1359.

Papadimitriou, P.; and Garcia-Molina, H. 2011. Data Leakage Detection. *IEEE Trans. Knowl. Data Eng.*, 23(1): 51–63.

Park, N.; Mohammadi, M.; Gorde, K.; Jajodia, S.; Park, H.; and Kim, Y. 2018. Data Synthesis based on Generative Adversarial Networks. *Proc. VLDB Endow.*, 11(10): 1071–1083.

Quiring, E.; Arp, D.; and Rieck, K. 2018. Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 488–502. IEEE.

Ratner, A.; Bach, S. H.; Ehrenberg, H. R.; Fries, J. A.; Wu, S.; and Ré, C. 2017a. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.*, 11(3): 269–282.

Ratner, A.; Bach, S. H.; Ehrenberg, H. R.; Fries, J. A.; Wu, S.; and Ré, C. 2020. Snorkel: rapid training data creation with weak supervision. *VLDB J.*, 29(2-3): 709–730.

Ratner, A. J.; Bach, S. H.; Ehrenberg, H. R.; and Ré, C. 2017b. Snorkel: Fast Training Set Generation for Information Extraction. In *SIGMOD*, 1683–1686.

Salem, A.; Zhang, Y.; Humbert, M.; Berrang, P.; Fritz, M.; and Backes, M. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *NDSS*.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *NeurIPS*, 6106–6116.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M. U.; and Sutton, C. 2017. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *NeurIPS*, 3308–3318.

Strack, B.; DeShazo, J. P.; Gennings, C.; Olmo, J. L.; Ventura, S.; Cios, K. J.; and Clore, J. N. 2014. Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. *BioMed Research International*.

Suciu, O.; Marginean, R.; Kaya, Y.; III, H. D.; and Dumitras, T. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *27th USENIX Security Symposium*, 1299–1316.

Whang, S. E.; and Garcia-Molina, H. 2013. Disinformation techniques for entity resolution. In *CIKM*, 715–720.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling Tabular data using Conditional GAN. In *NeurIPS*, 7333–7343.

Xu, L.; and Veeramachaneni, K. 2018. Synthesizing Tabular Data using Generative Adversarial Networks. *CoRR*, abs/1811.11264.

Yeom, S.; Giacomelli, I.; Fredrikson, M.; and Jha, S. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, 268–282. IEEE.

Zhu, C.; Huang, W. R.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *ICML*, 7614–7623.