Membership Inference Attacks Against Machine Learning Models via Prediction Sensitivity

Lan Liu, Yi Wang, Gaoyang Liu, *Member, IEEE*, Kai Peng, and Chen Wang, *Senior Member, IEEE*

Abstract—Machine learning (ML) has achieved huge success in recent years, but is also vulnerable to various attacks. In this article, we concentrate on membership inference attacks and propose Aster, which merely requires the target model's black-box API and a data sample to determine whether this sample was used to train the given ML model or not. The key idea of Aster is that the training data of a fully trained ML model usually has lower prediction sensitivities compared with that of the non-training data (i.e., testing data). Less sensitivity means that when perturbing a training sample's feature value in the corresponding feature space, the prediction of the perturbed sample obtained from the target model tends to be consistent with the original prediction. In this article, we quantify the prediction sensitivity with the Jacobian matrix which could reflect the relationship between each feature's perturbation and the corresponding prediction's change. Then we regard the samples with a lower as training data. Aster can breach the membership privacy of the target model's training data with no prior knowledge about the target model or its training data. The experiment results on four datasets show that our method outperforms three state-of-the-art inference attacks.

Index Terms—Machine learning, membership inference attack, prediction sensitivity, Jacobian matrix

1 Introduction

MACHINE learning (ML) has been booming over the past years due to the increasing computing power and various types of data. However, many researchers have discovered that ML models are vulnerable to many kinds of attacks recently, including adversarial attacks [1], [2], model stealing attacks [3], model inversion attacks [4], and privacy violation attacks [5].

In this article, we focus on *membership inference attacks* (MIA), whose goal is to determine whether a sample belongs to a given model's training set [6]. MIAs could seriously endanger a model's data security. For example, a bank releases an ML model that can predict people's credit rating. If we know that one person's profile was used to train this model, we could infer that this person is more likely to be a customer of this bank.

Most MIAs require prior knowledge to perform MIAs. A part of MIAs require the target model's structure [6], [7], [8] or the target model's parameters [9]. Some works require access to the model's training process such that they could obtain the training loss of the training data [10]. Besides, there are some works that require the training samples or the statistical information of the data distribution [11]. Nevertheless, to construct a practicable ML model, developers usually have to put great efforts, including data collection, data cleaning, model selection, and model training. So in order to protect their benefits, most ML models are deployed as a service with only black-box access. In practice, the Internet giants including Google AI Platform[1] and Amazon ML[2] also provide their only provide black-box API of the ML models that are deployed on their servers. Black-box API means that we can only send the input sample to the target model and receive the corresponding output prediction. Therefore, many MIA methods are usually not applicable to the black-box scenarios.

In this article, we present a novel MIA dubbed Aster, which merely requires the perturbation of the target record in its feature space and the black-box prediction interface of the target ML model. Consistent with existing MIAs [6], [8], [12], we only consider attacks against the prediction process in the entire ML pipeline and ignore the parts of data preprocessing and anomaly detection before sending the inputs to ML models. Aster is based on the observation that a trained ML model usually is lesser sensitive to the feature value perturbations on its training samples compared with the non-training samples. In general, with the training process going on, the ML model will have more and more confidence in the predictions of the training data. When the training process is finished, the trained model usually has high robustness to different inputs in the neighborhood of its training data, or in other works, the prediction sensitivity. Consequently, with respect to a fully trained ML model, the perturbation to training samples would not cause a significant change to the model's prediction output. However, the non-training samples would have a higher sensitivity with respect to the model that they did not participate on the contrary. As such, by leveraging the prediction sensitivity difference between the data that the target model trained on versus the model sees for the first time, Aster can perform the inference attacks.

Although the basic idea of Aster is not complicated, we still confront two major challenges when we implement it. The first challenge is how to quantify a sample's prediction sensitivity with respect to a given ML model. Therefore, we propose a simple method to quantify the prediction sensitivity: we first add a perturbation to the target sample and then derive the changes in the model's output. However, the output changes corresponding to different degrees of perturbations are difficult to compare directly. In order to achieve a unified measurement, we use the derivative of a model's prediction with respect to the input to capture the sensitivity. Since an ML model's input and output usually have a large number of dimensions, we obtain the model output's partial derivatives in respect of each feature and then combine all the derivatives into a matrix, which coincidentally is the Jacobian matrix, to measure the sensitivity of the target sample.

Another challenge we met is that how to launch the attack for a single sample. Since we do not have the ground truth for the target samples' membership property, Aster only can leverage the unsupervised algorithm to divide the samples into two groups. However, existing unsupervised clustering algorithms can not be performed on the training dataset with only one sample. To solve this problem, we manually generate some samples by adding noise to the original sample. The noisy samples are not likely to be used in the training process of the target models, so we could utilize

- *Lan Liu is with the National Engineering Research Center for Educational Big Data, Central China Normal University, Wuhan, Hubei 430079, China. E-mail: lanliu@mail.ccnu.edu.cn.*
- *Yi Wang, Kai Peng, and Chen Wang are with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: {wangyiee, pkhust}@hust.edu.cn, cwangwhu@gmail.com.*
- *Gaoyang Liu is with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. E-mail: liugaoyang@hust.edu.cn.*

Manuscript received 29 July 2021; revised 30 May 2022; accepted 2 June 2022. Date of publication 10 June 2022; date of current version 13 May 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 61872416, 62171189, and 62002104; and in part by the Key Research and Development Program of Hubei Province under Grant 2020BAB120.
(Corresponding author: Gaoyang Liu.)
Digital Object Identifier no. 10.1109/TDSC.2022.3180828

1. https://cloud.google.com/ai-platform
2. https://aws.amazon.com/machine-learning

these samples to fit the requirement of the clustering algorithm. The details would be present in Section 4.3.

Our major contributions are summarized as follows:

- We present Aster, an MIA against ML models merely requiring the black-box prediction access, based on the observation that a trained ML model usually is less sensitive to feature space's perturbation on its training samples.
- We leverage the Jacobian matrix, which is composed of the relationship between the feature values of the input sample and the output prediction of the target model, to capture a given sample's prediction concerning the target model, and design an inference attack on the basis of the sensitivity difference between the training and testing data.
- We evaluate the performance of Aster against four different types of ML models on four datasets and then compare it with three existing MIAs.[3] Merely with the black-box interface of the target models, Aster could achieve a mean precision of 0.601 and a mean recall of 0.775, which are much higher than those of existing methods.

2 Related Work

2.1 Membership Inference Attacks

The first MIA against ML models is proposed by Shokri *et al.* [6]. They built several local shadow models to mimic the prediction behavior of the target model and then make use of the shadow models' outputs to train a set of attack models. This work requires prior knowledge of both the target model's structure, training settings, or the training platform. Afterward, Salem *et al.* [12] showed that only one shadow model and one attack model would be enough to perform MIAs. However, this work still requires prior knowledge of the target model's training data. Li *et al.* [13] proposed an instance-probability attack. It trains several shadow models, and then extracts the membership feature with shadow models' predictions.

Besides leveraging shadow models, some works leverage other information of the target model to perform MIAs. For example, Yeom *et al.* [10] leveraged the average loss of all training data to perform the attacks. Their work requires all the training data. Wu *et al.* [9] computed the membership probability with the target model's parameters and an auxiliary dataset that contains samples from the dataset and ground truth membership label. Then they set a threshold to distinguish the members from the non-members of the training data. Nasr *et al.* [8] extracted a membership score for the target model based on the model's activations, predictions, and losses. This work requires prior knowledge of the target model's structure and parameters. Leino *et al.* [14] built their attack model by deriving a set of parameters describing the use of special features. This work requires the internal structure of the target model. Hui *et al.* [15] proposed an MIA, called BlindMI. Their attack probes the target model and extracts membership semantics via differential comparison.

Recent works focus on a more practical scenario in which the target model only outputs the predicted label. Choquette-Choo *et al.* [16] proposed the first label-only MIAs. Their attacks evaluate the robustness of a model's predicted labels under perturbations to obtain a fine-grained membership signal. Li *et al.* [17] also presented label-only MIAs. They proposed two types of decision-based attacks named transfer-attack and boundary-attack respectively. Transfer-attack works under the scenario that the adversary has a dataset (called shadow dataset) that comes from the same distribution as the target model's training set. Then they construct some shadow models to launch a membership inference attack locally. Boundary-attack does not require the shadow dataset. This attack adds noises to the target sample and attempts to change its label predicted by the target model. Then by measuring the amount of added noise, the adversary can whether the target sample was used to train the target model or not.

TABLE 1
Mean Jacobian Matrix Norm

Setting		Norms of Jacobian Matrix	
Dataset	Model	Training Data	Testing Data
Adult	NN	130.736	148.627
Adult	LR	139.405	159.848
Bank	NN	134.300	179.274
Bank	LR	143.877	147.601
MNIST	NN	13.346	35.33
MNIST	LR	99.623	126.383

2.2 Applications of Jacobian Matrix

Some works that discuss the security of machine learning leverage the Jacobian matrix. Clements *et al.* [18] found a way to insert a backdoor into a trained neural network to make it malfunction by making use of the Jacobian matrix. Jakubovitz *et al.* [19] proposed a novel method to improve the robustness of neural networks by using the Frobenius norm of the Jacobian of the network. Novak *et al.* [20] discussed the sensitivity and generalization in neural networks, they use the norm of the Jacobian matrix as the metric to measure the generalization of the networks. In other fields, some works use the Jacobian matrix to help to analyze their models.

Recently, Chen *et al.* [21] proposed a novel Jacobian-matrix-adaption (JMA) method for the tracking control of robot manipulators via the zeroing dynamics. Their solution based on the JMA method transforms the internal, implicit, and unmeasurable model information to the external, explicit, and measurable input-output relationships. Pope *et al.* [22] used the Jacobian matrix to analyze multispecies models. They calculate the Jacobian matrix of long-term steady-state catch by species concerning the fishing mortality relative to status quo levels on all species. Using this method is possible to compare different model estimations of fishing mortality rate changes needed to approach yield-related management goals.

Jacobian matrix has been widely deployed in many different fields. It helps to analyze how a system's output reacts when some changes occur to the input. In this article, we leverage the Jacobian matrix to measure the sensitivity for the target model to different samples. Then we compute norms of the Jacobian matrix to divide the target samples into the member and non-member groups.

3 Motivation

Aster leverages the observation that the Jacobian matrix of the training samples usually has a smaller norm compared with that of the non-training samples. Therefore, to solidify our observation and prove the feasibility of Aster, we conduct several experiments on three datasets in this section.

To study how the Jacobian matrix distributes differently for samples from the target model's training set and those that are not, we trained several target models with the standard training process. For every target model, we have 100 samples from the testing set and 100 samples from the training set. Then we approximate the Jacobian matrix norm for these samples(about approximation methods, please refer to Section 4) and compare the mean norm. The results are shown in Table 1. The results are perfectly consistent with our motivation. For every target model, the amplitude of norms varies for different target models, but the mean Jacobian

3. Our code is available at https://github.com/Wangyiiiiii/Aster

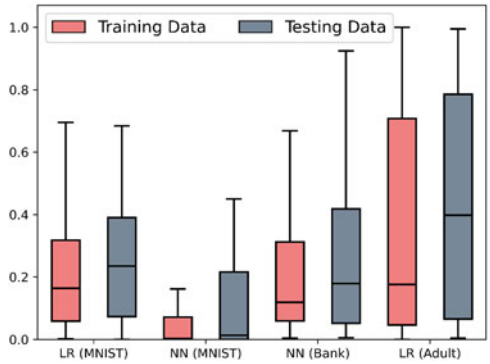

Fig. 1. The distribution of Jacobian matrix norm (The norms are scaled with the max for better visualization).

matrix norms of samples from the training set are strictly less than the mean jacobian matrix norms of samples from the testing set. Furthermore, we observe how the Jacobian matrix norm of samples is distributed for four different target models. The results are shown in Fig. 1. It shows that the distribution of the Jacobian matrix norms of samples from the training set are generally lower than that of testing set.

Consequently, from the preliminary experiments, we can see that the sample in a model's training set is less sensitive to perturbations. The prediction sensitivity can be captured by the Jacobian matrix of a given sample with respect to the ML model. In addition, we also find that the norm of the Jacobian matrix is consistent with the prediction sensitivity. As a consequence, we use the norm of the Jacobian matrix to measure the prediction sensitivity of the target sample.

4 Design of Aster

4.1 Threat Model

The goal of Aster is to determine whether a sample was used to train the target model or not. Aster does not presume any prior information about the target model or its training data. In other words, the adversary that we consider in Aster only has the black-box API of the target model. The details of our threat model are described below.

4.1.1 Target Model

In this article, the target model represents the victim whose training data is under attack. We concentrate on the classification ML models. For the target sample, we input it to the target model $\mathcal{M}$, and the target model will return the prediction probability vector. Each value in the vector represents the probability that the target sample is predicted to belong to the corresponding category by the target model. The sum of the probability vector should be exactly equal to 1.0. We formalize the above procedure as follows: $y = \mathcal{M}(\mathbf{x})$, where $\mathbf{x}$ is the input sample, and y is the predicted probabilities vector. $\mathcal{M}$ here represents the target model's black-box API.

4.1.2 Priori Knowledge of Aster

We consider a more practical adversary who only has the black-box access to the target model. This assumption limits our attack, that is, we can not obtain the information about target model's structure, type, parameters, training algorithm, and settings, as well as its training data's statistics. Therefore, the only information we can obtain from the target model is the prediction probability vector of a given input.

4.1.3 The Capability of Aster

The only interaction allowed between the adversary and the target model $\mathcal{M}$ is to query $\mathcal{M}$ with a sample $\mathbf{x}$ and then get the prediction output

$$\mathcal{M}(\mathbf{x}) = [y^1, y^2, \ldots, y^c, \ldots, y^{|C|}], \tag{1}$$

where C is the set of class labels that the target model can take, and y^c is the probability that the input sample belongs to class c. The only information the adversary can receive from the target model $\mathcal{M}$ is the prediction probability vector.

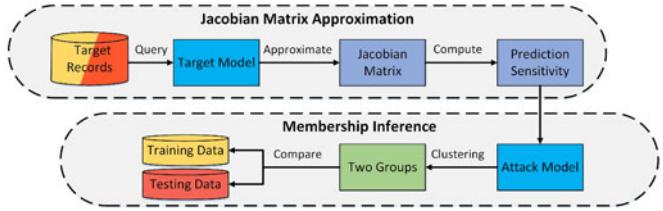

Fig. 2. Overview of aster.

4.1.4 The Goal of Aster

Given the target model $\mathcal{M}$ and a target sample $\mathbf{x}_t$, the adversary attempts to determine whether $\mathbf{x}_t$ is from $\mathcal{M}$'s training set or not

$$\mathcal{A}(\mathbf{x}_t, \mathcal{M}) \rightarrow \mathbf{In}/\mathbf{Out}. \tag{2}$$

$\mathcal{A}$ is the abstract for the adversary. It takes $\mathcal{M}$ and $\mathbf{x}_t$ as input and outputs two kinds of labels. **In** means that $\mathbf{x}_t$ is from $\mathcal{M}$'s training set while **Out** has the opposite meaning.

4.2 Methodology of Aster

Given a target sample $\mathbf{x}_t$ and the target model $\mathcal{M}$ with only black-box access, we are interested in whether $\mathbf{x}_t$ was used to train $\mathcal{M}$ or not. Aster first approximates the Jacobian matrix for $\mathbf{x}_t$ and derives the prediction sensitivity with respect to $\mathcal{M}$. Then Aster clusters the target samples according to the sensitivity and thus determines whether $\mathbf{x}_t$ is from $\mathcal{M}$'s training set or not. Therefore, there are mainly two steps in Aster to perform MIAs (c.f. Fig. 2): *(1) Jacobian Matrix Approximation* and *(2) Membership Inference*. Next, we will elaborate on each step.

4.2.1 Jacobian Matrix Approximation

An ML model can be regarded as a function $\mathcal{M} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that maps n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ to m-dimensional output $\mathbf{y} \in \mathbb{R}^m$. Then the Jacobian matrix of $\mathcal{M}$ is defined to be an m $\times$ n matrix, whose element located in ith row and jth column is $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$ ($i \in [1, 2, \ldots, n]$ and $j \in [1, 2, \ldots, m]$)

$$\mathbf{J}(\mathbf{x}; \mathcal{M}) = \begin{bmatrix} \frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}, \tag{3}$$

where $\mathbf{y} = \mathcal{M}(\mathbf{x})$. The input sample is $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, and the corresponding prediction is $\mathbf{y} = [y_1, y_2, \ldots, y_m]$. $\frac{\partial y_i}{\partial x_j}$ represents the relationship between the change of the input sample's ith feature value and the change of the prediction probability that this sample belongs to jth class.

From the definition of the Jacobian matrix, we can see that the matrix is made up of a series of first-order partial derivatives. Even we do not have the internal states of function $\mathcal{M}$, we still can approximate these derivatives calculating the numerical differentiation with the following equation:

$$\frac{\partial y_j}{\partial x_i} \approx \frac{\mathcal{M}(\mathbf{x} + \epsilon) - \mathcal{M}(\mathbf{x} - \epsilon)}{2\epsilon}, \tag{4}$$

where ϵ is a small value added to the ith feature value of the input sample.

For the target sample $\mathbf{x}_t$ whose membership property we are interested in, we add ϵ to (resp. minus ϵ from) the ith feature value of the target sample and get two modified samples. Then we query the target model with the two modified samples, and derive the partial derivatives of ith feature with respect to the target model: $\frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_i} = [\frac{\partial y_1}{\partial x_i}, \frac{\partial y_2}{\partial x_i} \ldots \frac{\partial y_m}{\partial x_i}]$. We repeat the above process for each feature in $\mathbf{x}$ successively and combine the partial derivatives into the Jacobian matrix.

Now that we have the approximation of the Jacobian matrix which is defined as $\mathbf{J}(\mathbf{x}; \mathcal{M})$ for clarity. Then we need to extract the prediction sensitivity of the target sample with respect to the target model. Following Novak *et al.* [20], we make use of L-2 norm of $\mathbf{J}(\mathbf{x}; \mathcal{M})$ to represent the prediction sensitivity for the target sample. For a $m \times n$ matrix $\mathbf{A}$, the L-2 norm of $\mathbf{A}$ can be computed by

$$\|A\|_2 = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}, \tag{5}$$

where i and j are the row number and the column number of the matrix element $a_{i,j}$, respectively.

4.2.2 Membership Inference

With the prediction of the target sample (i.e., the norm of the Jacobian matrix approximation $\mathbf{J}(\mathrm{x}; \mathcal{M})$), we can determine whether the target sample is from the target model's training set or not. Our work is based on the observation that ML models usually exhibit less sensitivity concerning the prediction behavior on the training data [20]. The other key observation is that the distribution of the L-2 norm of the Jacobian matrix between the training and testing data has been illustrated in the previous section. Based on the above preliminaries, we could find that the prediction sensitivity of samples from the training set is generally lower than that of the samples from the testing set.

It comes naturally that we can leverage an unsupervised clustering method to group a set of target records into 2 clusters and then determine the cluster with a lower mean sensitivity as the members of the $\mathcal{M}$'s training set. However, through our experiments, we find that Aster may perform even better when the cluster number is larger. Therefore, we try different numbers of clusters and set the number to 6 based on the average attack performance against different models and dataset. We compared several clustering algorithms including K-Means, Spectral Clustering, and DBSCAN. The Spectral Clustering obtains the best performance in our experiments, therefore we finally decided to use the spectral clustering algorithm to construct the attack clustering model.

When we implement Aster, we use a simple trick to get better performance. At the inference stage, we first cluster the samples into three or more groups, and then order the groups by the average norm. We try different numbers of clusters and record the corresponding attack performance of Aster. From the results, we find that the performance generally improves first and then declines as the number of clusters increases. Empirically, we decide to use 6 clusters. Finally, we predict that the groups with smaller average norm are from the target model's training set and others are not.

4.3 Discussion: Aster Against Single Target Sample

The above methodology of Aster is designed under the scenario that the adversary has a suspicious dataset consisting of multiple target samples grabbed from the training and testing set of the target model. With this suspicious dataset, Aster could infer the membership property of the target records by clustering the Jacobian matrix norms (i.e., prediction sensitivity) into two groups. If the adversary can merely get a single target sample, our attack could fail since existing unsupervised clustering algorithms can not be performed on the training dataset with only one sample.

To tackle this problem, we first duplicate the target sample multiple times and then add some random noises to each duplication. The majority of the noised duplications never participate in the training process of the target model. Therefore, we first generate the local samples around the target sample. We select a part of the features of $\mathbf{x}_t$ uniformly at random and perturb the value of these features. For the numerical features, we directly add random noise to the original feature values. For the categorical features, we randomly choose another label from the range that this feature can take. After perturbing the duplications, we calculate the Jacobian matrix for both $\mathbf{x}_t$ and its perturbed duplications.

Next, we can directly reuse the same clustering method in Section 4.2 to infer the membership property of $\mathbf{x}_t$. If the target sample is from the target model's training set, the Jacobian norms of perturbed duplications should be larger than the Jacobian norm of the target sample. Else if the target sample is not from the target model's training set, then some perturbed duplications may get close to the samples from the target model's training set. The Jacobian norm of the target sample is more likely to be divided into the group with a larger mean Jacobian norm. At the end, we use the same criteria as discussed in Section 4.2.2 to determine the target sample's membership property.

5 Performance Evaluation

5.1 Experiment Setup

5.1.1 Datasets

To evaluate the performance of Aster, we use four public benchmark datasets in this article. These datasets include UCI Adult,[4] Purchase,[5] MNIST,[6] and Bank.[7] Because Purchase dataset does not have any class label, we adopt the labeling method of the works of Shokri *et al.* [6] and Salem *et al.* [12]. Specifically, we leverage the K-Means algorithm to cluster the samples into 100 groups and each cluster corresponds to a style of purchase paternal. Then the cluster results would serve as the ground truth label of Purchase dataset. In our experiments, we randomly select 10,000 samples from each dataset to constitute the training data for the target model.

5.1.2 Target Models

To evaluate the performance of our MIA thoroughly, we use four different ML algorithms to train the target models, including random forest (RF), neural networks (NN), logistic regression (LR), and support vector machine (SVM). Since we assume the target model is deployed as black-box, we only make use of a prediction interface of the target model once its training process is finished.

5.1.3 Evaluation Metrics

In our experiments, we mainly make use of two standard metrics *precision* and *recall* to evaluate the performance of Aster. *Precision* presents the proportion of the data samples predicted as members of the training dataset that are indeed in the target model's training set. *Recall* presents the fraction of the training samples that we can correctly infer as the training set's samples. In other words, precision measures the attack accuracy while recall measures the attack coverage. We also used *F1-score* when we study the impact of different choices for epsilon. *F1-score* could convey the balance between the precision (i.e., attack accuracy) and the recall (i.e., attack coverage).

4. https://archive.ics.uci.edu/ml/datasets/Adult
5. https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data
6. http://yann.lecun.com/exdb/mnist/
7. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

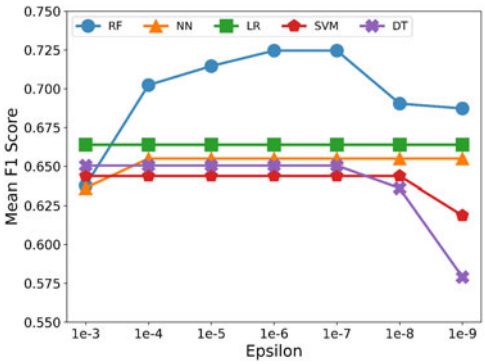

Fig. 3. The choice of epsilon.

5.1.4 Comparison Methods

We compare our attack with the following three MIAs:

Shokri et al. [6] This MIA first trains several shadow models which have identical algorithms and structure as the target model to simulate target models' prediction behavior. Then it trains multiple attack models with the shadow models' outputs to perform MIA.

ML-Leaks. [12] This MIA leverages several different ML algorithms to train a set of sub-models, and then combines these sub-models as one shadow model. In the end, it trains an attack model the same way as Shokri *et al.*'s work.

Nasr et al. [8] This MIA leverages an auto-encoder for extracting the membership property of the target data and then trains an unsupervised cluster model to infer which sample is in the target model's training set.

In the following experiments, we optimize the parameters of models for all the methods. We also set the number of samples that are from the target model's training is the same as the number of samples that are from the testing set, thus the baseline of inference attack accuracy is 50%.

5.2 Choice of Epsilon

One key step of our MIA is to derive the Jacobian matrix approximation of the target sample with respect to the target model. According to Eq. (4), we know that different ϵ will affect the results of the approximation of the Jacobian matrix and then degree the performance of Aster. Therefore, we need to select a proper value of ϵ to derive the approximation of the Jacobian matrix with respect to the given ML model. Therefore, we perform the inference attacks against 20 different target models with different values of ϵ, ranging from 1e-3 to 1e-9. In order to achieve a balance between accuracy and recall, we use the F1 score as the metric to determine the value of ϵ.

The experiment results are shown in Fig. 3. For RF models, the F1 score of our attacks increases from 0.637 to 0.724 as ϵ varies from 1e-3 to 1e-6, and then keeps stable when ϵ decreases to 1e-7. Finally, the F1 score decreases to 0.687 when ϵ is set to 1e-9. For NN models, the F1 score increases from 0.636 to 0.654 as ϵ decreasing from 1e-3 to 1e-4, and then stays all the same for the rest value choices of ϵ. For LR models, the attack performance of Aster is not affected by the different choice of ϵ at all. The F1 score can achieve a mean value of 0.663 for the whole time. For SVM models, Aster could achieve a mean F1 score of 0.643 as ϵ varying from 1e-3 to 1e-8. Then the F1 score decreases to 0.618 when we set ϵ to 1e-9. Finally, for DT models, Aster could achieve a mean F1 score of 0.650 when ϵ varies from 1e-3 to 1e-7.

According to the experiment results in this part, we find that when the value of ϵ equals to 1e-6, Aster could achieve the acceptable performance. As a consequence, we uniformly set ϵ to 1e-6 in the rest experiments.

5.3 Performance of Aster

We first evaluate the performance of Aster, and the experiment results are summarized in Table 2. The objective of Aster is to infer the members of the target model's training data. Therefore, we evaluate the performance of Aster by executing it against four types of ML models. In addition, we also compare our attack with three existing works of MIAs.

For the four kinds of target models, we can see that Aster performs better than Shokri *et al.*'s work, ML-Leaks, and Nasr *et al.*'s work. Aster achieves a mean attack precision of 0.601 among all target models, which is 9.8%, 8.3%, and 9.5% higher than the three compared methods, respectively. As for the attack recall, the mean recall of Aster is 0.775, which is 13.7%, 16.2%, 19.2% higher than the three comparison methods, respectively.

Specifically, we first discuss the precision results achieved by Aster and comparison methods. For RF models, Aster achieves a mean precision of 0.609, which is better than all comparisons. It is 5.1% higher than Shokri *et al.*'s work, 7.8% higher than ML-Leaks, and 11.2% higher than Nasr *et al.*'s work. As for NN models, the mean precision of Aster is 0.604, which is also the highest among the comparisons. It is 12.1%, 8.9%, 13.6% higher than others, respectively. Third, for LR models, Aster achieves 0.595 mean precision, which is 13.6%, 8.9%, 4.3% higher than each comparison respectively. For SVM models, the mean precision of Aster is 0.594. That means Aster is also the highest among the four attack methods. And it surpasses the comparison methods by 8.3%, 7.6%, and 6.6%, respectively.

As for the recall metric, the details of the experiment results are elaborated as follows. For RF models, Aster achieves a mean attack recall of 0.899, which is 35.9%, 27.9%, and 19.9% higher than the three comparison methods respectively. As for NN models, the

TABLE 2
Attack Performance Comparisons

Target Model	LR				RF				NN				SVM			
Dataset	Adult	Bank	MNIST	Purchase	Adult	Bank	MNIST	Purchase	Adult	Bank	MNIST	Purchase	Adult	Bank	MNIST	Purchase
Metric	**Attack Precision**															
Shokri *et al.*	0.571	0.476	0.625	0.562	0.375	0.536	0.500	0.520	0.456	0.511	0.538	0.333	0.500	0.545	0.500	0.500
ML-Leaks	0.575	0.487	0.562	0.500	0.481	0.222	0.533	**0.636**	0.454	0.333	0.571	**0.666**	0.523	0.520	0.542	0.486
Nasr *et al.*	0.583	0.511	0.428	0.466	0.522	0.478	0.578	0.388	0.500	0.477	**0.733**	0.500	**0.692**	0.500	0.485	0.433
Aster	**0.615**	**0.585**	**0.633**	**0.605**	**0.714**	**0.571**	**0.612**	0.520	**0.678**	**0.585**	0.533	0.586	0.642	**0.571**	**0.558**	**0.605**
Metric	**Attack Recall**															
Shokri *et al.*	0.798	0.410	0.605	0.361	0.117	**0.881**	0.800	0.520	**0.836**	0.922	**0.843**	0.040	**1.000**	0.964	0.562	0.562
ML-Leaks	0.922	0.762	0.722	0.800	0.521	0.800	**0.962**	0.279	0.800	0.779	0.643	0.800	0.881	**1.000**	**0.762**	0.723
Nasr *et al.*	0.839	0.921	0.477	0.563	**0.921**	0.878	0.439	0.282	0.762	0.836	0.439	**0.762**	0.360	0.121	0.682	0.517
Aster	**0.960**	**0.960**	**0.758**	**0.919**	0.800	0.800	0.760	**0.522**	0.760	**0.958**	0.637	0.678	0.722	0.475	**0.762**	**0.922**

TABLE 3
Performance for Single Sample

Target Model Dataset	Model	Metric precision	Metric recall
Bank	NN	0.55	0.88
Bank	LR	0.534	0.92
Bank	SVM	0.5	0.8
MNIST	NN	0.484	0.64
MNIST	LR	0.58	0.72
MNIST	SVM	0.586	0.6

mean attack recall of Aster is 0.72 and also is the best among the four methods. Aster performs better than the comparisons by 14.0%, 26.0%, and 8.9% respectively concerning the recall metric. For LR models, Aster also achieves the best attack recall compared with the comparison methods. The mean recall of Aster is 0.76, which is 9.9%, 35.0%, and 6.0% higher than the comparisons. When it comes to SVM models, there is a remarkable difference in the experiment results. The recall of ML-Leaks is the highest among all the MIAs. Then the second highest is Shokri's work, which is 0.77. Our method comes to the third-highest, which is 0.72. And the last one is the attack proposed by Nasr *et al.*, which achieves a mean recall of 0.42.

5.4 Performance of Aster Against Single Target Sample

In this section, we evaluate the performance of Aster when facing only one target sample. We randomly select 1,000 samples from the target model's training data and testing data, respectively. Then we perform Aster against every sample one by one. To be specific, we duplicate every sample 49 times. Our experiments are performed on Bank and MNIST datasets, and the results are shown in Table 3.

From Table 3, we can see that by adding random noise to the sample, we still get good results on these target models. For bank dataset and MNIST dataset, the precision performance for NN, LR, SVM models we get are 0.55, 0.534, 0.5 and 0.484, 0.58, 0.586 respectively. The mean precision is 0.539, which is only 3.3% lower than the mean precision of the original multiple samples scene in Table 2. For recall performance, the mean for a single sample attack is 0.759. *Overall, by adding extra steps of duplication and perturbation process, Aster can obtain the ability to handle inference attacks against a single target sample.*

Naturally, there is a corner case when attacking against one single sample which is the target sample, and the perturbed samples are all non-members. In such a case, we can find that the member and non-member imbalance in our target dataset would greatly affect the performance of Aster, and this still needs more explorations in the future.

5.5 Impact of Number of Classes

Naturally, the number of output classes of the target model contributes to how much membership information the target model reveals. The larger number of classes, the more signals about the internal state of the model are available to Aster. To quantify the impact that the class number of the target models has on the performance of Aster, we further do a set of experiments with Purchase dataset.

Specifically, we first adopt K-Means algorithm to cluster the samples in Purchase dataset into 2, 10, 20, 50, and 100 clusters respectively and then assign each sample with the cluster results as the class label. Then we train a series of NN models on Purchase dataset with a different number of classes. After that, we execute Aster against these models successively. The experiment results are shown in Fig. 4.

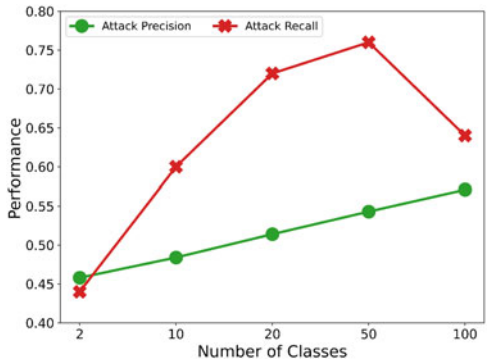

Fig. 4. The impacts of the number of classes (Purchase).

From the results, we can see that models with fewer classes leak less information about their training data and thus the performance of Aster gets better as the number of classes increasing. To be specific, the attack precision increases from 0.458 to 0.571 as the number of classes increases from 2 to 100. On the other hand, the attack recall also increases as the number of classes increases except for the number of classes increasing to 100. One possible reason is that models with larger class number need to remember more about their training data, thus they are easier to get a high overfitting level. A high overfitting level represents that the prediction of these models will have a high tolerance of the training data's perturbation and thus the training sample has a much lower sensitivity compared with the testing sample. In general, models with more output classes are more vulnerable to our attack.

5.6 Norms of Jacobian Matrix for Different Classes

In this section, we also conduct a set of experiments to study the sensitivity difference across different classes of training and testing data on MNIST dataset. Specifically, in order to quantify the sensitivity, we derive the norms of the Jacobian matrix for samples from different classes. We train the target models on MNIST dataset, since it has 10 classes that correspond to the handwritten digits from 0 to 9. The experiment results are summarized in Table 4.

We derive the Jacobian matrix for each class in both training and testing data with the numerical approximation, and then calculate the mean of L-2 norm for each class. From the experiment results, we can see that although the norm varies from one class to another, it stays at the same magnitude. We can also see that for most classes (except for classes 4 and 9), the norm of the training data is lower than that of the testing data.

5.7 White-Box versus Black-Box

We also perform additional experiments to show that our inference attack is valid under both white-box and black-box settings. To be clear, by white-box we mean that we have the access to the inner parameters of the target models. That makes it possible for us to derive the precise formula to compute the Jacobian matrix for a certain sample. We train multiple NN models on three datasets, including iris,[8] wine,[9] and MNIST. The results are shown in Table 5. Iris and wine are two commonly used toy benchmark datasets. Iris dataset consists of 50 samples from each of three species of Iris. Each samples contains four features: the length and the width of the sepals and petals, in centimeters. Based on the combination of these features, the researcher could classify the Iris into different species. The wine dataset contains the results of a chemical analysis of wines grown in a specific area of Italy. Three wines are represented in the 178 samples, with 13 features of chemical analyses results.

8. https://archive.ics.uci.edu/ml/datasets/iris
9. https://archive.ics.uci.edu/ml/datasets/wine

TABLE 4
L-2 Norm of Jacobian Matrix for LR Models (MNIST Dataset)

Model	Class	0	1	2	3	4	5	6	7	8	9
LR	**Training Data**	71.135	87.790	81.941	130.467	125.947	144.659	88.853	110.061	124.679	148.031
LR	**Testing Data**	79.603	94.831	108.494	134.736	121.572	151.904	102.015	115.983	132.483	146.594
SVM	**Training Data**	12.878	9.477	17.472	24.390	31.880	41.818	17.138	22.261	27.140	39.748
SVM	**Testing Data**	13.423	9.889	32.854	28.462	21.019	48.340	24.186	24.627	29.513	40.658

TABLE 5
White-Box versus Black-Box

Setting		L-2 Norm of Jacobian Matrix	
Dataset	Category	White-box	Black-box
Iris	Training Data	117.775	117.753
Iris	Testing Data	146.825	146.904
Wine	Training Data	43.368	43.368
Wine	Testing Data	53.717	54.524
MNIST	Training Data	40.914	40.930
MNIST	Testing Data	48.386	48.387

From the table we can see that our black-box Jacobian matrix approximation is very close to the precise result in three different datasets for both the training data and testing data. That means our approximation method performs precisely even without any information about the target model.

6 Conclusion

In this article, we have explored feature space perturbation and presented Aster, a novel MIA against ML models with only black-box API. We observe that the sensitivity of the training data with respect to a fully trained ML model is generally lower than that of the non-training data. To reveal the sensitivity, we make use of the Jacobian matrix to measure the relationship between the target model's prediction and the target sample's feature value, and perform MIAs by comparing the sensitivity values of different data samples. Experiments on various types of target models and datasets show that even without any prior knowledge about the target model and the statistical information of its training set, Aster can outperform other MIAs. We hope our work could illustrate the risk of membership privacy in real-world ML models and facilitate the emergence of effective defenses of MIAs. However, Aster would not perform well for models that have similar sensitivity derived from Aster to the training and testing data. We thus plan to explore more essential relationship between an ML model and its training data. We will also consider performing MIAs against the whole pipeline of ML models including data pre-processing.

References

[1] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *ACM Digit. Threats: Res. Pract.*, 2021, doi: 10.1145/3469659.

[2] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1332–1349.

[3] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 36–52.

[4] X. Wang, R. Hou, Y. Zhu, J. Zhang, and D. Meng, "NPUFort: A secure architecture of DNN accelerator against model inversion attack," in *Proc. 16th ACM Int. Conf. Comput. Front.*, 2019, pp. 190–196.

[5] L. T. Phong and T. T. Phuong, "Privacy-preserving deep learning via weight transmission," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 11, pp. 3003–3015, Nov. 2019.

[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[7] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jegou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5558–5567.

[8] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[9] B. Wu *et al.*, "Characterizing membership privacy in stochastic gradient langevin dynamics," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 04, pp. 6372–6379, 2020.

[10] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp.*, 2018, pp. 268–282.

[11] G. Liu, C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng, "SocInf: Membership inference attacks on social media health data with machine learning," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 5, pp. 907–921, Oct. 2019.

[12] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/ml-leaks-model-and-data-independent-membership-inference-attacks-and-defenses-on-machine-learning-models/

[13] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in supervised learning via generalization gap," in *Proc. ACM CODASPY*, 2021, pp. 5–16.

[14] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *Proc. USENIX Secur.*, 2020, pp. 1605–1622.

[15] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–17.

[16] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 1964–1974.

[17] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proc. ACM CCS*, 2021, pp. 880–895.

[18] J. Clements and Y. Lao, "Backdoor attacks on neural network operations," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2018, pp. 1154–1158.

[19] D. Jakubovitz and R. Giryes, "Improving DNN robustness to adversarial attacks using jacobian regularization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 514–529.

[20] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: An empirical study," 2018, *arXiv:1802.08760*.

[21] D. Chen, Y. Zhang, and S. Li, "Tracking control of robot manipulators with unknown models: A jacobian-matrix-adaption method," *IEEE Trans. Ind. Inform.*, vol. 14, no. 7, pp. 3044–3053, Jul. 2018.

[22] J. G. Pope *et al.*, "Comparing the steady state results of a range of multi-species models between and across geographical areas by the use of th jacobian matrix of yield on fishing mortality rate," *Fisheries Res.*, vol. 209, pp. 259–270, 2019.