PAPER Special Section on Information and Communication System Security

Model Inversion Attacks for Online Prediction Systems: Without Knowledge of Non-Sensitive Attributes*

Seira HIDANO[†a)], ***Nonmember***, **Takao MURAKAMI**[††], ***Member***, **Shuichi KATSUMATA**[††,†††], ***Nonmember***, **Shinsaku KIYOMOTO**[†], ***and*** **Goichiro HANAOKA**[††], ***Members***

SUMMARY The number of IT services that use machine learning (ML) algorithms are continuously and rapidly growing, while many of them are used in practice to make some type of predictions from personal data. Not surprisingly, due to this sudden boom in ML, the way personal data are handled in ML systems are starting to raise serious privacy concerns that were previously unconsidered. Recently, Fredrikson et al. [USENIX 2014] [CCS 2015] proposed a novel attack against ML systems called the *model inversion attack* that aims to infer *sensitive* attribute values of a target user. In their work, for the model inversion attack to be successful, the adversary is required to obtain two types of information concerning the target user prior to the attack: the output value (i.e., prediction) of the ML system and all of the *non-sensitive* values used to learn the output. Therefore, although the attack does raise new privacy concerns, since the adversary is required to know all of the non-sensitive values in advance, it is not completely clear how much risk is incurred by the attack. In particular, even though the users may regard these values as non-sensitive, it may be difficult for the adversary to obtain all of the non-sensitive attribute values prior to the attack, hence making the attack invalid. The goal of this paper is to quantify the risk of model inversion attacks in the case when non-sensitive attributes of a target user are not available to the adversary. To this end, we first propose a general model inversion (GMI) framework, which models the amount of auxiliary information available to the adversary. Our framework captures the model inversion attack of Fredrikson et al. as a special case, while also capturing model inversion attacks that infer sensitive attributes without the knowledge of non-sensitive attributes. For the latter attack, we provide a general methodology on how we can infer sensitive attributes of a target user without knowledge of non-sensitive attributes. At a high level, we use the data poisoning paradigm in a conceptually novel way and inject malicious data into the ML system in order to modify the internal ML model being used into a *target* ML model; a special type of ML model which allows one to perform model inversion attacks without the knowledge of non-sensitive attributes. Finally, following our general methodology, we cast ML systems that internally use linear regression models into our GMI framework and propose a concrete algorithm for model inversion attacks that does not require knowledge of the non-sensitive attributes. We show the effectiveness of our model inversion attack through experimental evaluation using two real data sets.

key words: *black box, model inversion, data poisoning, online ML systems*

1. Introduction

With the rapid growth of artificial intelligence (AI), countless numbers of IT companies using machine learning (ML) algorithms as a service have emerged in the past few years. In many of these IT services, ML systems** are used to provide some type of predictions using personal data as input. To name a few of these successful ML systems, we have product recommendation [2]–[4], destination prediction [5]–[7], and personalized medicine [8]–[10]. However, while these systems provide meaningful service to the users, it also raises serious privacy concerns. In particular, the data being used by the ML systems are often times personal data that include sensitive information, which the users may want to keep secret (e.g., purchase history of sensitive items, home addresses, genetic markers). Thus, analyzing the vulnerability of ML systems in terms of privacy has recently caught much attention both in academic and business.

Specifically, in this paper, we consider the privacy aspects of *online* ML systems [11], [12]; a type of ML system where the data becomes available in sequential order and is used to update the current internal prediction model accordingly. Online learning is a common technique used when it is computationally infeasible to learn against the entire dataset in one shot or when the data arrives dynamically and is not known prior to learning the model. Online prediction systems are well motivated by the latter reason, since user data are typically unavailable for learning the model until the users have joined the prediction system***. Due to the practical appeal of online ML systems, it has been attracting research targeting privacy exploits [13], [14] and references therein). We note that since the usage scenario of online ML systems are extremely versatile, there are many attack surface, and hence, a "valid attack" depends greatly on the objective of the adversary and the application in one's mind.

Recently, Fredrikson et al. [15], [16] proposed a new privacy attack on ML systems called *model inversion attacks*. The attack aims to expose the *sensitive* attributes of a target user. At a high level, an adversary given as input an

Manuscript received November 14, 2017.
Manuscript revised May 8, 2018.
Manuscript publicized August 22, 2018.

[†]The authors are with the KDDI Research, Inc., Fujimino-shi, 356–8502 Japan.
[††]The authors are with the National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, 135–0064 Japan.
[†††]The author is with the University of Tokyo, Kashiwa-shi, 277–8561 Japan.

*A preliminary version of this paper was presented at the 15th International Conference on Privacy, Security and Trust (PST 2017), Calgary, Alberta, Canada, August 2017 [1].

a) E-mail: se-hidano@kddi-research.jp

DOI: 10.1587/transinf.2017ICP0013

**In this paper, we define the ML system as a system that internally uses an ML model (e.g., linear regression model, decision tree, neural network) to offer a service to users. We provide its formal definition in Sect. 3.1.

***Throughout the paper, we assume ML systems to be online, unless stated otherwise.

output value of the ML model (e.g., recommended item or place), searches the input values corresponding to the output value inversely (i.e., narrows down candidates for the input values). As long as the number of candidates are not large, the adversary will be able to infer sensitive attribute values with high probability. Fredrikson et al. first proposed a model inversion attack for linear regression models [15] and subsequently proposed an model inversion attack for non-linear ML models [16].

However, in the model inversion attack of Fredrikson et al. [15], [16], the adversary is required to obtain all of the *non-sensitive* attribute values of the target user, which are used as input to the ML model. In other words, if the ML model used non-sensitive attributes along with sensitive attributes to make the predictions, the adversary must obtain all of the non-sensitive attribute values to perform the model inversion attack. Examples of non-sensitive attributes may include demographic information (e.g., age, sex, occupation), purchase histories of commodities, and mobility traces around sightseeing places[†]. One important observation is that even though the users may consider such attributes as non-sensitive, it may not be always easy for an adversary to obtain all of them.

For example, suppose an ML system that recommends items to users based on their age, sex, occupation, and purchase history. Specifically, input of the ML model is the age, sex, occupation, and purchase history, and the output of the ML model is the recommended item. Assume the users consider their purchase history as sensitive information, and the demographic information (i.e., age, sex, occupation) as non-sensitive information. Now, suppose the users disclose the recommended item to the public (e.g., via SNS). In this case, even if the users consider age, sex, and occupation as non-sensitive attributes, it is not a simple nor easy task for the adversary to obtain all of these information (e.g., the users may disclose these information only to their close friends via SNS; the users may simply not disclose them for some other reason). Therefore, in such a case the adversary will not be able to execute the model inversion attack of [15], [16], since the attack requires a priori the non-sensitive attributes. Thus, ideally speaking (from the adversaries' point of view), the model inversion attack should work without the knowledge of the non-sensitive attributes of the target users.

The goal of this paper is to quantify the risk of model inversion in the case when non-sensitive attributes of a target user are not available to the adversary. To achieve this goal, we propose a new framework for model inversion attacks that enables the adversary to infer sensitive attribute values without the knowledge of the non-sensitive attributes. More specifically, we focus on *online prediction systems*, in which users provide their personal data, and a prediction model is updated online based on the personal data. Examples of such systems include item recommendation systems [2]–[4] and destination prediction systems [5]–[7]. Then we propose a *general model inversion* (GMI) framework for prediction systems, which models the amount of auxiliary information available to the adversary prior to the attack. The GMI framework captures the previous model inversion attack of [15], [16] as a special case. More importantly, the proposed framework also allows us to model a new type of model inversion attack for prediction systems that can be carried out without the knowledge of non-sensitive attributes,

One may think that it is very difficult (or even impossible) to infer the sensitive attribute values of a user without the knowledge of the non-sensitive attribute values (i.e., from only the output value). However, we try to bypass this difficulty by utilizing the fact that the prediction systems can be modified. Note that this is where we crucially rely on the fact that the ML system is *online*. Specifically, we borrow ideas from the *data poisoning* paradigm [17]–[28]. Data poisoning has been widely studied mainly to increase the prediction error by injecting malicious training data into the ML model. It has also been studied to boost/reduce the popularity of specific items in item recommendation systems [28]. In this paper, we utilize the data poisoning technique in a novel way to achieve our goal. Specifically, we inject malicious training data into the ML model to modify the ML model into a target ML model *without* much degradation of the prediction accuracy. Here we try to keep the prediction accuracy unchanged to avoid detection. The target ML model is chosen in such a way that we can perform model inversion attacks without the knowledge of non-sensitive attributes.

Based on the GMI framework, we also propose an algorithm for model inversion attacks against linear regression models. The linear regression model is one of the most basic ML models and is also used for solving the cold start problem of recommendation systems [29]. We show the effectiveness of our proposed algorithm through experimental evaluation using two real data sets: "How Americans Like Their Steak" [30] and MovieLens 1M Dataset [31].

1.1 Our Contributions

Contributions of this paper can be summarized as follows:

- We propose a new framework called the *general model inversion* (GMI) framework. This framework models the amount of auxiliary information available to the adversary. It includes the model inversion attack in [15], [16] as a special case. It also enables a new type of model inversion attack that infers sensitive attributes without the knowledge of non-sensitive attributes by modifying the ML model into a target ML model via data poisoning (Sect. 3).

[†]Note that each user may have a different set of sensitive attributes and non-sensitive attributes. For example, Alice may want to keep her age and occupation as secret, but does not mind if her whole purchase history is disclosed to someone. On the other hand Bob may make his age and occupation to the public via SNS, but want to keep his purchase history of sensitive items (e.g., medicine, pornography) as secret.

- We then propose an algorithm for model inversion attacks against linear regression models. The experimental results using two real data sets showed that our model inversion attack, which does not require the knowledge of non-sensitive attributes, performs almost the same way as the model inversion attack in [15], [16], which requires the knowledge of non-sensitive attributes. In addition, the prediction accuracy was not much degraded by injecting malicious data. These results show that the adversary can infer sensitive attributes without the knowledge of non-sensitive attributes by utilizing data poisoning (Sects. 4 and 5).

1.2 Paper Organization

The rest of this paper is organized as follows: In Sect. 2, we review the model inversion attack in [15], [16] and the previous work on data poisoning. In Sect. 3, we propose the general model inversion (GMI) framework. In Sect. 4, we propose an algorithm for model inversion attacks against linear regression models. In Sect. 5, we report the experimental results. In Sect. 6, we conclude the paper.

2. Related Work

In this section, we provide a brief overview on the model inversion attacks introduced by Fredrikson et al. [15], [16], and existing work concerning data poising for ML systems.

2.1 Model Inversion Attacks

In 2014 and 2015, Fredrikson et al. [15], [16] proposed a new type of attack on (not necessarily online) ML systems aiming to expose sensitive information of users, called *model inversion attacks*. At a high level, an adversary exciting the model inversion attack first obtains an output value y concerning the target user from the ML system. Then, by using the ML system for its unintended purpose, the adversary infers the sensitive attribute values $\mathbf{x}$ that were used to learn the output value y. In the following, we provide a more detailed explanation.

Let f be an ML model (e.g., linear regression model, decision tree, neural network) that makes a prediction using D attributes as input, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$ be an input attribute space, and $\mathcal{Y}$ be an output space. Furthermore, we use lower case letters $\mathbf{x} = (x_1, \ldots, x_D) \in \mathcal{X}$ and $y \in \mathcal{Y}$ to indicate input attributes and output values of the ML model f, respectively. Here, for some $T \in [D]$, let $(x_1, \ldots, x_T)$ be the *sensitive* attributes and $(x_{T+1}, \ldots, x_D)$ be the *non-sensitive* attributes; sensitive attributes are those attributes that the users wish to keep secret. Finally, let $\mathbf{p} = (p_1, \ldots, p_D)$ be the prior distributions of the input attributes and err be a Gaussian error function. Then, the model inversion attack of Fredrikson et al. is defined by an algorithm $\mathsf{ModelInversion}(f, y, (x_{T+1}, \ldots, x_D), \mathbf{p}, \mathsf{err}) \to (x_1, \ldots, x_T)$. Here, we like to point out that the attack of Fredrikson et al., requires the adversary to know the non-sensitive attribute values $(x_{T+1}, \ldots, x_D)$ prior to the attack. In words, the adversary narrow downs the candidate sensitive attribute values $(x_1, \ldots, x_T)$ by using the ML model f, the output value y, the non-sensitive attributes $(x_T, \ldots, x_D)$, and the prior distribution $\mathbf{p}$. In particular, in case the sensitive attribute value is not uniquely defined, the $\mathsf{ModelInversion}$ algorithm picks the most likely value by using the maximum a posteriori probability (MAP) estimator with prior distributions $\mathbf{p}$ of the input attributes and the Gaussian error function err. We provide the concrete algorithm of the model inversion attack of Fredrikson et al. [15], [16] in Algorithm 1 (note that Fredrikson et al. [15], [16] defined err as a likelihood; the posterior probability $r_{\hat{\mathbf{x}}}$ is proportional to the product of the likelihood $\mathsf{err}(y, f(\hat{\mathbf{x}}))$ and the prior probabilities $p_1(\hat{x}_1), \cdots, p_D(\hat{x}_D)$).

Algorithm 1 $\mathsf{ModelInversion}(f, y, (x_{T+1}, \ldots, x_D), \mathbf{p}, \mathsf{err})$

for $\hat{x}_1, \ldots, \hat{x}_T$ **in** $\mathcal{X}_1 \times \cdots \times \mathcal{X}_T$ **do**
  $\hat{\mathbf{x}} \leftarrow (\hat{x}_1, \ldots, \hat{x}_T, x_{T+1}, \ldots, x_D)$
  $r_{\hat{\mathbf{x}}} \leftarrow \mathsf{err}(y, f(\hat{\mathbf{x}})) \cdot \prod_{i=1}^{D} p_i(\hat{x}_i)$
end for
return $\arg\max_{\hat{\mathbf{x}}} r_{\hat{\mathbf{x}}}$

In their pioneering work, Fredrikson et al. applied the model inversion attack to linear regression models, and in the subsequent work [16] they furthered their attack to non-linear models such as decision trees and neural networks. In both of their works, they used actual data sets to prove heuristically through experiments that the success probability of recovering the sensitive attributes improve when the adversary is given non-sensitive attributes, apposed to when adversaries are restricted to only information of the prior distribution $\mathbf{p}$. Subsequently, Wu et al. [32] gave a formalization of the model inversion attack of Fredrikson et al. Their formalization only captures the attack scenario where the adversary is given all the knowledge of the non-sensitive attributes of the target user.

Even if some or all of the non-sensitive attributes are unknown, the model inversion attack of Algorithm 1 can be executed by regarding the non-sensitive attributes as sensitive attributes; $T = D$. However, such a method can result in low attack accuracy. In the MAP estimator for the model inversion attack, it is implicitly assumed that input attributes are independent and identically distributed, as it is hard to specify the joint distribution (see [15] for the details). If there is some strong correlation among the attributes, the prior distributions $\mathbf{p} = (p_1, \ldots, p_D)$ may convey little useful information. Intuitively, in this case, the more attributes the adversary would like to recover, the lower the attack success rate will be, due to the lack of information on the correlation. In fact, we show in Sect. 5 that the attack success rate really decreases with increase in the number of recovered attributes, and that the attack success rate is very low when $T = D$. Therefore, we leverage another approach, i.e., data poisoning, for avoiding the reduction of the attack success rate when non-sensitive attributes are unknown.

2.2 Poisoning to ML systems

Data poisoning is a type of attack that aims to degrade the performance of ML models. It has attracted much attention in the past decade with a long line of interesting research [17]–[28]. In the early stage, the security of data poisoning was theoretically analyzed in terms of probably approximately correct (PAC) learning [17]–[19]. Recently, data poisoning attacks for particular ML models, rather than general ML models, have been extensively studied. Among them, poisoning attacks on support vector machines (SVMs), investigated by Biggio et al. in [20], [21], are one of the recent representative studies. They proposed a data poisoning algorithm that drastically degrades the performance of SVMs by injecting only a few malicious data into the training data. Very recently, Li et al. [28] proposed a poisoning attack on collaborative filtering following a similar approach as Biggio et al. On the other hand, Cao et al. [33] proposed a data poisoning approach on ML models, where data are maliciously removed from the training data, rather than injected. While many methods for data poisoning are known, the common goal shared by all methods is to degrade the performance of the ML model, or to boost/reduce the popularity of specific items in the collaborative filtering system. We note that in the following, we depart from this common goal and use data poisoning with a different objective in mind; we use data poisoning to alter an ML model into a desired ML model, for which we can conduct our model inversion attack. We refer the readers to other survey papers for further details on poisoning attacks [22]–[27].

3. General Model Inversion Framework

In this section, we propose a *general model inversion* (GMI) framework, which is inspired by the seminal studies of Fredrikson et al. [15], [16]. Our framework captures the scenarios of the previous studies, and also those (previously uncaptured) scenarios where the adversary is *not* in possession of non-sensitive attributes. It should be noted that although non-sensitive attributes are not kept hidden from the users, this does not necessarily mean that such information is disposable to the adversary. In many scenarios, it may be more natural to assume that the adversary will not be able to collect all the non-sensitive attributes prior to the attack. In particular, ideally, we would want the model inversion attack to succeed without requiring to know what the non-sensitive attributes are.

Below, we first define the ML system we intend to attack and propose our formalization of GMI attacks. Then, we provide detailed discussions on the connection with previous studies and how each phase can be implemented.

3.1 Attack Target: Online Prediction Systems

In this paper, we define an *ML system* as a system that internally uses an *ML model* to offer services to users. Here, ML models are learning models such as linear regression models, decision trees, neural networks and so on. In our work, we consider a particular type of ML system called the *prediction system* PredSys as an attack target for our model inversion attack (see Fig. 1). As we have mentioned in Sect. 1, prediction systems are widely deployed for practical applications such as product recommendation systems [2]–[4] and destination prediction systems [5]–[7]. The ML model associated with the prediction system is the *prediction model* $f : \mathcal{X} \to \mathcal{Y}$ that deterministically maps a D-dimensional attribute vector $\mathbf{x}$ in the feature space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$ to a value y in the range $\mathcal{Y}$. An attribute vector $\mathbf{x}$ is divided into sensitive attributes $(x_1, \ldots, x_T)$ and non-sensitive attributes $(x_{T+1}, \ldots, x_D)$ for some $T \in [D]$. We say a prediction model is used for classification (resp. regression) depending on whether $\mathcal{Y}$ is finite (resp. infinite).

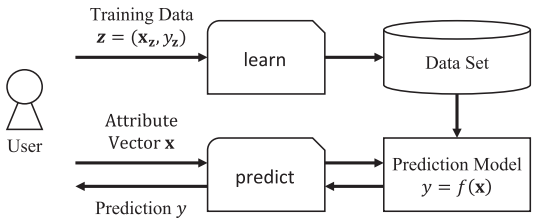

Fig. 1 Prediction System.

Furthermore, we assume the prediction model f is obtained by supervised learning; the model f is learned from some training data set $\{\mathbf{z}_i\}_{i\in[N]} = \{(\mathbf{x}_{\mathbf{z}_i}, y_{\mathbf{z}_i})\}_{i\in[N]}$, where N denotes the number of data. To distinguish training data from the actual input/output pair $(\mathbf{x}, y)$ of the prediction model, we use subscripts $(\mathbf{x}_\mathbf{z}, y_\mathbf{z})$ to emphasize that the data are training data. Specifically, in our work, we are interested in *online* supervised learning. This is a particular type of learning method suited for situations where data become available in a sequential manner, which is often the case in the aforementioned applications for prediction systems. Formally, we define a prediction system PredSys for a class of predicate models $\mathcal{F}$:

Definition 1 (Prediction System): Let $\mathcal{F}$ be a class of prediction models, $\mathcal{X}$ be a feature space and $\mathcal{Y}$ be the range of the prediction models. Then, a *prediction system* PredSys for $\mathcal{F}$ is defined by the following two functions (learn, predict):

- learn$(\mathbf{z} = (\mathbf{x}_\mathbf{z}, y_\mathbf{z}); f) \to \perp$: This is the *learning* function that takes the training data $\mathbf{z} = (\mathbf{x}_\mathbf{z}, y_\mathbf{z})$ as input and updates internally the current prediction model $f \in \mathcal{F}$ of the prediction system PredSys to a prediction model $f_{\mathsf{upt}} \in \mathcal{F}$. It terminates by outputting a null symbol $\perp$.
- predict$(\mathbf{x}; f) \to \mathbf{y}$: This is the *predict* function that takes an attribute vector $\mathbf{x} \in \mathcal{X}$ as input and outputs $y = f(\mathbf{x}) \in \mathcal{Y}$ as the prediction, where $f \in \mathcal{F}$ is the current prediction model of the prediction system PredSys.

We provide some intuition on the above definition. Both functions learn and predict are publicly executable,

where they are both implicitly parameterized by the current prediction model f of the prediction system PredSys. This is expressed by the "; f" notation in the input. In other words, one can execute learn and predict without knowledge of the current prediction model f. This captures many of the real-world prediction systems where one does not know exactly how machine learning algorithms are deployed internally in the systems. As an example, even if we do not know the current prediction model f being used in the prediction system (say a recommendation system), we can add a new user with some initial attributes to the recommendation system to modify the inner prediction model f (using learn) or we can find out what kind of items are recommended by the model by creating artificial users (using predict).

3.2 General Model Inversion Attack

We formalize the notion of general model inversion (GMI) attacks. Our formalization captures the scenario where an adversary tries to infer the *sensitive* attributes $(x_1, \ldots, x_T)$ of the input attributes $\mathbf{x} \in \mathcal{X}$ when provided with only the corresponding output value $y \in \mathcal{Y}$ of the prediction model f, i.e., $f(\mathbf{x}) = y$. The distinguishing feature between the attack of Fredrikson et al. [15], [16] and ours is that the adversary is not required to know the values of the *non-sensitive* attributes $(x_{T+1}, \ldots, x_D)$ of the input attributes $\mathbf{x}$ to execute the attack.

At a high level, our approach is to modify the prediction model f of the prediction system PredSys into a prediction model f_{tgt} such that we can recover the sensitive attributes without knowledge of the non-sensitive attributes. To this end, we take full advantage of the public functions (learn, predict) of the prediction system PredSys. Recall, that (learn, predict) can be run by anybody, including the adversary. Namely, the function predict is used to infer parameters of the current prediction model f of PredSys (e.g., coefficient vectors of linear regression models) and the function learn is used to modify the current prediction model f into a *target* prediction model f_{tgt}, which we know how to exploit. In the following, we present the formalization of our GMI framework and provide a detailed discussion of the syntax and notions used. The overview of our GMI framework is depicted in Fig. 2.

Definition 2 (General Model Inversion Attacks): Let $\mathcal{F}$ be a class of prediction models, $\mathcal{X}$ be a feature space, and $\mathcal{Y}$ be the range of the prediction models. Let $\mathcal{F}_{\mathsf{tgt}} \subset \mathcal{F}$ be a class of target prediction models. Then, a (general) model inversion attack for the prediction system PredSys = (learn, predict) for $\mathcal{F}$ is defined by the tuple of three algorithms (Setup, Poisoning, ModelInversion):

- $\mathsf{Setup}(\mathsf{init}) \to (f_{\mathsf{cur}}, \mathsf{par}_{\mathsf{sys}})$: This is the *setup* algorithm that takes an initialization parameter init as input and outputs the current prediction model $f_{\mathsf{cur}} \in \mathcal{F}$ and some system-specific parameter $\mathsf{par}_{\mathsf{sys}}$ of the prediction system PredSys.

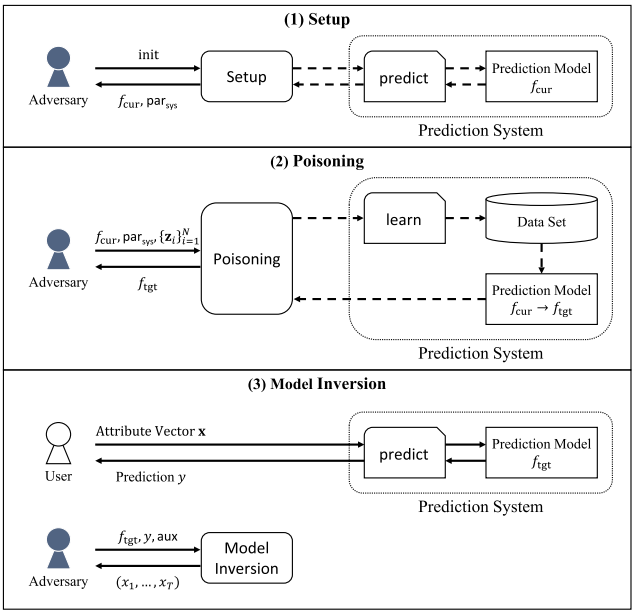

Fig. 2 GMI framework. Solid lines connected to the setup, poisoning, or model inversion algorithm represent input/output data. The dashed lines connected to these algorithms represent the interaction between these algorithms and the prediction system.

- $\mathsf{Poisoning}(f_{\mathsf{cur}}, \mathsf{par}_{\mathsf{sys}}, \{\mathbf{z}_i = (\mathbf{x}_{\mathbf{z}_i}, y_{\mathbf{z}_i})\}_{i=1}^N) \to f_{\mathsf{tgt}}$: This is the *poisoning* algorithm that takes the current prediction model f_{cur} of the prediction system PredSys, a system-specific parameter $\mathsf{par}_{\mathsf{sys}}$, and a set of malicious data $\{\mathbf{z}_i = (\mathbf{x}_{\mathbf{z}_i}, y_{\mathbf{z}_i})\}_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$ for some positive integer N as input, and updates the current prediction model $f_{\mathsf{cur}} \in \mathcal{F}$ to some target prediction model $f_{\mathsf{tgt}} \in \mathcal{F}_{\mathsf{tgt}}$. Finally, it outputs f_{tgt}.
- $\mathsf{ModelInversion}(f_{\mathsf{tgt}}, y, \mathsf{aux}) \to (x_1, \ldots, x_T)$: This is the *model inversion* algorithm that takes a prediction model $f_{\mathsf{tgt}} \in \mathcal{F}_{\mathsf{tgt}}$, some output $y \in \mathcal{Y}$ of a user, and some auxiliary information aux as input, and outputs a tuple of sensitive attributes $(x_1, \ldots, x_T)$.

We assume that all algorithms implicitly take the description of PredSys = (learn, predict) as input, i.e., each algorithm is allowed to run learn and predict as a subprotocol.

3.3 Discussion of the Formalization

We discuss the formalization of our GMI attack in detail.
- Algorithm Setup: The setup algorithm is used to infer the current prediction model f_{cur} and some system-specific parameter $\mathsf{par}_{\mathsf{sys}}$ of the prediction system PredSys. The initialization parameter init is some value specific to the attacking environment. The system-specific parameter $\mathsf{par}_{\mathsf{sys}}$ is some parameter that is used internally by function learn, e.g., the learning rate or the step size used by to learn the prediction model. In case algorithm Poisoning does not require $\mathsf{par}_{\mathsf{sys}}$ as input, it can be simply set as $\mathsf{par}_{\mathsf{sys}} = \perp$. One way to construct the Setup algorithm is by running the

function predict provided by PredSys. Specifically, we iteratively run the function predict on adaptively-chosen inputs $\mathbf{x} \in \mathcal{X}$ to obtain the corresponding output $y \in \mathcal{Y}$, and narrow the candidates of the prediction model using the input/output pair. A concrete example is provided in the next section. Finally, although algorithm Setup may be run multiple times, we name it Setup to emphasize that it only needs to be run once for our application in mind.

- Algorithm Poisoning: The poisoning algorithm is used to modify the current prediction model f_{cur} of the prediction system PredSys into a target prediction model f_{tgt}, i.e., a prediction model that we know how to model invert. Note that we do not necessarily know which target prediction model $f_{\mathsf{tgt}} \in \mathcal{F}_{\mathsf{tgt}}$ it is modified to. Our high level approach is to use the function learn provided by the prediction system PredSys to inject malicious data into the current prediction model f_{cur}. We require f_{cur} and $\mathsf{par}_{\mathsf{sys}}$ as input to know the initial state of the prediction system.

- Algorithm ModelInversion: The model inversion algorithm is used to infer the sensitive attributes of a target user. The auxiliary information aux signifies the amount of resource an adversary has. For example, if aux is the non-sensitive attributes $(x_{T+1}, \cdots, x_D)$ and the prior distribution $\mathbf{p}$, then we recover the model of Fredrikson et al. [15], [16]. In particular, in case the adversary is not in possession of the non-sensitive attributes, then aux is simply set as the prior distribution $\mathbf{p}^{\dagger}$. We may formalize any resource an adversary has by defining aux as an appropriate function $g(\mathbf{x})$ for attributes $\mathbf{x} \in \mathcal{X}$.

4. Application to Linear Regression Models

In this section, we provide a (general) model inversion attack for prediction systems using the class of linear regression models (i.e., the class of prediction models $\mathcal{F}$). To this end, we introduce the class of target linear regression models (i.e., the class of target prediction models $\mathcal{F}_{\mathsf{tgt}}$) that enable inference of sensitive attribute values only from the output value. We also provide a concrete description of our GMI attack defined by the three algorithms (Setup, Poisoning, ModelInversion).

4.1 Prediction Systems Using Linear Regression Models

Following the formalization provided in Sect. 3.1, we define the type of prediction system PredSys = (learn, predict) we intend to attack. We begin by defining the class $\mathcal{F}_{\mathsf{Lin}}$ of linear regression models. Note that this defines the function predict of the prediction system PredSys.

Definition 3 (Class of Linear Regression Models): Let $\mathcal{X}$ be the feature space and $\mathcal{Y}$ be the output space. A *linear regression model* $f_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$ is defined by following degree-$(D+1)$ polynomial:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\top}\mathbf{x} = w_0 + w_1 x_1 + \cdots w_D x_D, \tag{1}$$

where $\mathbf{x} = (1, x_1, \cdots, x_D)^{\top} \in \mathcal{X}$ is an attribute vector and $\mathbf{w} \in \mathbb{R}^{D+1}$ is the regression coefficients. Then, a *class of linear regression models* $\mathcal{F}_{\mathsf{Lin}}$ is defined as follows:

$$\mathcal{F}_{\mathsf{Lin}} = \{f_{\mathbf{w}} \mid \forall \mathbf{w} \in \mathbb{R}^{D+1}\}. \tag{2}$$

Next, we specify the function learn we consider for the prediction system PredSys. As discussed in Sect. 3.1, we consider online sequential supervised learning to be one of the most natural learning algorithms for training the prediction model with our application in mind. In particular, we assume the stochastic gradient descent (SGD) algorithm [34] is used as the learning algorithm.

Definition 4 (Stochastic Gradient Descent): Let f be a prediction model in the class of linear regression models $\mathcal{F}_{\mathsf{Lin}}$. Given a training data set $\{\mathbf{z}_i\}_{i\in[N]} = \{(\mathbf{x}_{\mathbf{z}_i}, y_{\mathbf{z}_i})\}_{i\in[N]}$, the stochastic gradient descent (SGD) algorithm updates the regression coefficients $\mathbf{w}$ of f using the following equation:

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \eta(y_{\mathbf{z}_i} - \mathbf{w}^{(i)\top}\mathbf{x}_{\mathbf{z}_i})\mathbf{x}_{\mathbf{z}_i},\ 1 \le i \le N, \tag{3}$$

where η is the learning rate and $\mathbf{w}^{(1)}$ is initialized to an arbitrary value.

This completes the specification of the type of prediction system PredSys = (learn, predict) we consider. Finally, in this paper, we utilize the root mean square error (RMSE) as a measure of the prediction accuracy of linear regression models.

Definition 5 (Root Mean Square Error): Let f be a model in the class of linear regression models $\mathcal{F}_{\mathsf{Lin}}$ and let $\{(\mathbf{x}_i, y_i)\}_{i\in[N]}$ be the data set for evaluation, where N is the number of data. Then, the root mean square error (RMSE) of f is defined as:

$$R_f = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2. \tag{4}$$

4.2 Overview of Our Approach

We provide an overview of our approach before providing the full description of our concrete (general) model inversion attack. We first define the class of target prediction models and provide a high level idea of our data poisoning algorithm Poisoning.

4.2.1 Target Prediction Model

Here, we define the class of target linear regression models $\mathcal{F}_{\mathsf{tgt}} \subset \mathcal{F}_{\mathsf{Lin}}$ (i.e., the target prediction model). Recall from Sect. 3.2 that a target prediction model allows one to infer the sensitive attributes $(x_1, \cdots, x_T)$ of the input attributes $\mathbf{x} = (1, x_1, \cdots, x_D)^{\top\dagger\dagger}$ while only using the knowledge of

†We assume that the prior distribution $\mathbf{p}$ can always be obtained in some particular format as done in previous studies (see for example [15], [16]).

††Note that we slightly alter the syntax to cope with the bias term of the linear regression model; the feature space is defined as $\mathcal{X} = \{1\} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$.

the corresponding output y. Intuitively, we would like to define the class of $\mathcal{F}_{\mathsf{tgt}}$ so that the sensitive attributes can be uniquely (and efficiently) retrieved from the output value y. In case of linear regression models, one way to approach this is by ensuring that none of the non-sensitive attributes contribute to the prediction model. In particular, the following target prediction model captures this intuition:

Definition 6 (Class of Target Prediction Models): Let $\mathcal{F}_{\mathsf{Lin}}$ be a class of linear regression models. Let $\mathbf{x} = (1, x_1, \ldots, x_T, \ldots, x_D)^\top$ be a vector in the feature space $\mathcal{X}$ where $(x_1, \ldots, x_T)$ are the sensitive attributes and $(x_{T+1}, \ldots, x_D)$ are the non-sensitive attributes for some $T \in [D]$. Then, a *target prediction model* $f_{\mathbf{w}} \in \mathcal{F}_{\mathsf{Lin}}$ is defined as the following polynomial:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} = w_0 + w_1 x_1 + \cdots w_T x_T, \tag{5}$$

where $\mathbf{w}$ is the regression coefficients in the space $\mathbb{R}^{T+1} \times \{0\}^{D-T}$ (i.e., $\mathbf{w} = (w_0, \cdots, w_T, 0, \cdots, 0)^\top$). Then, a *class of target prediction models* $\mathcal{F}_{\mathsf{tgt}} \subset \mathcal{F}_{\mathsf{Lin}}$ is defined as follows:

$$\mathcal{F}_{\mathsf{tgt}} = \{f_{\mathbf{w}} \mid \mathbf{w} \in \mathbb{R}^{T+1} \times \{0\}^{D-T}\}. \tag{6}$$

We like to note that even though we defined the target model $\mathcal{F}_{\mathsf{tgt}}$ as described above, there may be other candidates for the target model. One of the criteria to keep in mind is that the target model should be defined so that the output value y has a small preimage.

4.2.2 Poisoning Algorithm

Here, we provide an overview of our data poisoning algorithm Poisoning. Due to our definition of the target prediction model $f_{\mathsf{tgt}} \in \mathcal{F}_{\mathsf{tgt}}$, we must inject malicious data $\mathbf{z} = (\mathbf{x}_{\mathbf{z}}, y_{\mathbf{z}})$ into the prediction system PredSys so that the regression coefficients $(w_{T+1}, \cdots, w_D)$ corresponding to the non-sensitive attributes of the current prediction model $f_{\mathsf{cur}} \in \mathcal{F}_{\mathsf{Lin}}$ are modified to zero. Intuitively, we would like to inject as few malicious data as possible into the prediction system PredSys to accomplish this goal.

In the following, denote $\mathbf{z}_{\mathsf{DB}} = \{\mathbf{z}_i\}_{i=1}^N$ as an actual data set for some positive integer N. For the time being, assume $\mathbf{z}_{\mathsf{DB}}$ is provided. We will discuss later on how to create this data set. To create the malicious data, we first sample $\mathbf{z}$ from $\mathbf{z}_{\mathsf{DB}}$, where $\mathbf{z} = (\mathbf{x}_{\mathbf{z}}, y_{\mathbf{z}})$, $\mathbf{x}_{\mathbf{z}} = (1, \mathbf{x}_{\mathsf{sen}}, \mathbf{x}_{\mathsf{non\text{-}sen}})^\top$ and $\mathbf{x}_{\mathsf{sen}}$ (resp. $\mathbf{x}_{\mathsf{non\text{-}sen}}$) denotes the sensitive attributes $(x_{\mathbf{z}_1}, \cdots, x_{\mathbf{z}_T})$ (resp. non-sensitive attributes $(x_{\mathbf{z}_{T+1}}, \cdots, x_{\mathbf{z}_D})$). Next, we view $\mathbf{x}_{\mathsf{non\text{-}sen}}$ as variables[†] and solve the following $D - T$ simultaneous equations for $\mathbf{x}_{\mathsf{non\text{-}sen}}$.

$$0 = w_j + \eta(y_{\mathbf{z}} - \mathbf{w}^\top \mathbf{x}_{\mathbf{z}}) x_{\mathbf{z}_j} \ (T+1 \le j \le D), \tag{7}$$

where η is the learning rate, which is part of the system-specific parameter $\mathsf{par}_{\mathsf{sys}}$ (outputted by algorithm Setup) provided to algorithm Poisoning as input.

Looking at Eq. (3), we can see that Eq. (7) is solving for $\bar{\mathbf{x}}_{\mathsf{non\text{-}sen}}$ that would set the regression coefficients of the non-sensitive attributes in the next iteration of the SGD algorithm to zero. Therefore, if we were allowed to inject the malicious data $\{(1, \mathbf{x}_{\mathsf{sen}}, \bar{\mathbf{x}}_{\mathsf{non\text{-}sen}})^\top, y_{\mathbf{z}}\}$ into the prediction system PredSys, we would be able to modify the current prediction model f_{cur} into a target prediction model f_{tgt} with a malicious data point. However, in general this may not be possible, since the prediction system PredSys may not accept the values of $\bar{\mathbf{x}}_{\mathsf{non\text{-}sen}}$ as input. Namely, if each attribute space had a predefined range, i.e., $\mathcal{X}_j = [x_j^{\min}, x_j^{\max}]$, we first check whether $\bar{x}_{\mathbf{z}_j}$ falls within this interval. If not, we set the value to either $x_j^{\min}$ or $x_j^{\max}$ depending on whether $\bar{x}_{\mathbf{z}_j}$ is larger than the maximum value or not. Otherwise, $\bar{x}_{\mathbf{z}_j}$ will be accepted as a valid input to the prediction system PredSys.

Now, we discuss the aforementioned issue on how to create an actual data set $\mathbf{z}_{\mathsf{DB}}$. One of the simplest ways of creating this data set is to generate it from scratch. Assuming the prior distribution $\mathbf{p}$ for the attribute vectors is given, we can sample $\mathbf{x}$ from it and simply run $y \leftarrow \mathsf{predict}(\mathbf{x})$. For example, in case of recommendation systems, this procedure corresponds to creating new dummy users. Another way is to use publicly available data sets that may have some correlations with the data sets we want to acquire. Specifically, we may not obtain the actual data set we desire, it may be possible to format the obtained data set into the desired data set by piecing it together. Continuing on with the example of recommendation systems, even if we cannot obtain full information on who likes what other kinds of items, we can easily collect information on what kind of items are often bought together. Finally, another viable approach is for adversaries to respectively collect actual data from their friends, and collude and pool together the data they have collected. This approach is especially effective when the required number of malicious data is small (e.g., tens, a hundred), as in our experiments in Sect. 5, and/or the data are easily collected through SNS.

Finally, although it is not required, it would be preferable to maintain the prediction accuracy of the target prediction model f_{tgt} compared to the original prediction model f_{cur}. This additional restriction allows to evade detection of a poisoning attack from the prediction system PredSys while conducting the GMI attack. Interestingly, in the next section, we see that for some data sets the prediction accuracy does not degrade as much as anticipated.

4.3 Algorithm Construction

In this section, we provide the full description of our model inversion attack (Setup, Poisoning, ModelInversion) for the prediction system PredSys = (learn, predict) for the class of linear regression models $\mathcal{F}_{\mathsf{Lin}}$.

(1) Algorithm Setup

In order to construct the poisoning algorithm Poisoning based on the strategy discussed in Sect. 4.2, we first need to infer the regression coefficients $\mathbf{w}$ of the current prediction

[†]One can think as discarding the actual values $\mathbf{x}_{\mathsf{non\text{-}sen}}$ sampled from $\mathbf{z}_{\mathsf{DB}}$.

Algorithm 2 Setup($\mathbf{z}_{\mathsf{DB}}$)

function estimate_coeffs($\mathcal{X}$)
  for $j = 0$ **to** D **do**
    $\mathbf{x} \xleftarrow{\$} \mathcal{X}$
    $y \leftarrow \mathsf{predict}(\mathbf{x})$
    $\mathbf{eq}[j] \leftarrow y = \mathbf{w}^\top \mathbf{x}$
  end for
  $\mathbf{w} \leftarrow \mathsf{solve}(\mathbf{eq}, \mathbf{w})$
  return $\mathbf{w}$
end function

$\mathbf{w} \leftarrow \mathsf{estimate\_coeffs}(\mathcal{X})$
$\mathbf{z} \xleftarrow{\$} \mathbf{z}_{\mathsf{DB}}$
$\perp \leftarrow \mathsf{learn}(\mathbf{z})$
$\mathbf{w}' \leftarrow \mathsf{estimate\_coeffs}(\mathcal{X})$
$\mathsf{eq} \leftarrow w_1' = w_1 + \eta(y_z - \mathbf{w}^\top \mathbf{x}_\mathbf{z}) x_{\mathbf{z}_1}$
$\eta \leftarrow \mathsf{solve}(\mathsf{eq}, \eta)$
return ($f_{\mathsf{cur}} := f_{\mathbf{w}'}$, $\mathsf{par}_{\mathsf{sys}} := \eta$)

model $f_\mathbf{w} \in \mathcal{F}_{\mathsf{Lin}}$ and the learning rate η of the SGD algorithm (see Eq. (3)). Algorithm 2 is the complete description of our algorithm Setup, where the initialization parameter init is set as $\mathbf{z}_{\mathsf{DB}}$. Note that the symbol $\xleftarrow{\$}$ denotes the operation of sampling uniformly at random from a given set. **eq** denotes an array storing an equation in each entry, and $\mathbf{eq}[j]$ denote the j-th entry (i.e., equation). $\mathsf{solve}(\mathbf{eq}, \mathbf{w})$ is an algorithm that takes an array **eq** of equations and a variable $\mathbf{w}$ as input, and solves the simultaneous equations for $\mathbf{w}$. Note that solving systems of linear equations can be done efficiently using standard algorithms such as Gaussian elimination.

Algorithm Setup relies heavily on the functions predict and learn of the prediction system PredSys to estimate the prediction model f_{cur} (i.e., the regression coefficients $\mathbf{w}$) and the system-specific value $\mathsf{par}_{\mathsf{sys}}$ (i.e., the learning rate η). First, to calculate the regression coefficients $\mathbf{w}$, the algorithm calls the function predict $D+1$ times to generate $D+1$ simultaneous equations based on Eq. (1), and computes the coefficients $\mathbf{w}$ by solving the simultaneous equations running $\mathsf{solve}(\mathbf{eq}, \mathbf{w})$. Second, the learning rate η can be calculated from the regression coefficients $\mathbf{w}$ and $\mathbf{w}'$ of the prediction model before and after the update, respectively. After inferring the values of $\mathbf{w}^{(i)}$, the algorithm injects some training data $\mathbf{z}$ to the prediction system PredSys using the function learn, and then further infers the values of $\mathbf{w}'$. Then, η can be easily calculated using the first equation of Eq. (3). Note that the choice of which equation to use is arbitrary.

(2) Algorithm Poisoning

Algorithm 3 is the complete description of our data poisoning algorithm Poisoning for the prediction system PredSys = (learn, predict), where $f_{\mathsf{cur}} \in \mathcal{F}_{\mathsf{Lin}}$ is the current prediction model (outputted by the Setup algorithm). (list).any denotes the set of all the elements stored in list and clear_variable(var) denotes a function that clears whatever was set in the variable var. The rest of the notations are defined the same as in Algorithm 2.

Algorithm 3 Poisoning($f_{\mathsf{cur}} = f_\mathbf{w}$, $\mathsf{par}_{\mathsf{sys}} = \eta$, $\mathbf{z}_{\mathsf{DB}}$)

while $0 \notin (w_{T+1}, \ldots, w_D)$.any **do**
  $(\mathbf{x}_\mathbf{z}, y_\mathbf{z}) \xleftarrow{\$} \mathbf{z}_{\mathsf{DB}}$
  $\mathsf{clear\_variable}(x_{\mathbf{z}_{T+1}}, \ldots, x_{\mathbf{z}_D})$
  for $j = T+1$ **to** D **do**
    $\mathbf{eq}[j] \leftarrow 0 = w_j + \eta(y_\mathbf{z} - \mathbf{w}^\top \mathbf{x}_\mathbf{z}) x_{\mathbf{z}_j}$
  end for
  $\bar{x}_{\mathbf{z}_{T+1}}, \ldots, \bar{x}_{\mathbf{z}_D} \leftarrow \mathsf{solve}(\mathbf{eq}, (x_{\mathbf{z}_{T+1}}, \ldots, x_{\mathbf{z}_D}))$
  for $j = T+1$ **to** D **do**
    if $\bar{x}_{\mathbf{z}_j} > x_j^{\max}$ **then**
      $\bar{x}_{\mathbf{z}_j} \leftarrow x_j^{\max}$
    else if $\bar{x}_{\mathbf{z}_j} < x_j^{\min}$ **then**
      $\bar{x}_{\mathbf{z}_j} \leftarrow x_j^{\min}$
    end if
  end for
  $\mathbf{z}_{\mathsf{poison}} \leftarrow ((1, x_{\mathbf{z}_1}, \ldots, x_{\mathbf{z}_T}, \bar{x}_{\mathbf{z}_{T+1}} \ldots, \bar{x}_{\mathbf{z}_D})^\top, y_\mathbf{z})$
  $\perp \leftarrow \mathsf{learn}(\mathbf{z}_{\mathsf{poison}})$
  for $j = 0$ **to** D **do**
    $w_j \leftarrow w_j + \eta(y_{\mathbf{z}_{\mathsf{poison}}} - \mathbf{w}^\top \mathbf{x}_{\mathbf{z}_{\mathsf{poison}}}) x_{\mathbf{z}_{\mathsf{poison},j}}$
  end for
  $\mathbf{w} \leftarrow (w_0, w_1 \cdots, w_D)^\top$
end while
return $f_{\mathsf{tgt}} = f_\mathbf{w}$

Given an actual data set $\mathbf{z}_{\mathsf{DB}}$, the algorithm first samples $\mathbf{z} = ((x_{\mathbf{z}_1}, \ldots, x_{\mathbf{z}_T}, \ldots, x_{\mathbf{z}_D})^\top, y_\mathbf{z})$ from $\mathbf{z}_{\mathsf{DB}}$. Then (viewing the non-sensitive attributes $\mathbf{x}_{\mathsf{non\text{-}sen}} = (x_{\mathbf{z}_{T+1}}, \ldots, x_{\mathbf{z}_D})$ as variables) it solves the simultaneous equations of Eq. (7) for $\mathbf{x}_{\mathsf{non\text{-}sen}}$, and replaces the non-sensitive attribute values of $\mathbf{z}$ with the acquired solution $\bar{\mathbf{x}}_{\mathsf{non\text{-}sen}} = (\bar{x}_{\mathbf{z}_{T+1}}, \ldots, \bar{x}_{\mathbf{z}_D})$. In case the solutions fall outside the allowed domain $[x_j^{\min}, x_j^{\max}]$ of each attribute space $\mathcal{X}_j$, the algorithm replaces the values with the maximum value $x_j^{\max}$ or the minimum value $x_j^{\min}$ of each domain. Finally, the malicious data $\mathbf{z}_{\mathsf{poison}}$ is injected into the prediction system PredSys using the function learn. Here, one important observation is that by running function learn we update the current prediction model f. However, nonetheless, we can calculate the updated prediction model f on our own; we do not have to run the Setup algorithm for the next iteration. This process is iteratively run until $w_{T+1} = \cdots = w_D = 0$. In the actual experiment, we set a maximum number of iterations as a hyper-parameter to specify how much poisoning data we are willing to inject to the prediction model PredSys.

(3) Algorithm ModelInversion

Algorithm 4 is the complete description of our model inversion algorithm. We like to emphasize that we do not require the non-sensitive attributes $(x_{T+1}, \ldots, x_D)$ of the target user; the auxiliary information aux is only the prior distribution $\mathbf{p}$ and the Gaussian error function err. As in Fredrikson

Algorithm 4 ModelInversion($f_{\mathsf{tgt}} = f_\mathbf{w}$, y, $\mathsf{aux} = (\mathbf{p}, \mathrm{err})$)

for $\hat{x}_1, \ldots, \hat{x}_T$ **in** $\mathcal{X}_1 \times \cdots \times \mathcal{X}_T$ **do**
  $\hat{\mathbf{x}} \leftarrow (1, \hat{x}_1, \ldots, \hat{x}_T)^\top$
  $r_{\hat{\mathbf{x}}} \leftarrow \mathsf{err}(y, f(\hat{\mathbf{x}})) \cdot \prod_{i=1}^T p_i(\hat{x}_i)$
end for
return $\arg\max_{\hat{\mathbf{x}}} r_{\hat{\mathbf{x}}}$

et al. [15], [16], we construct the MAP estimator using the prior distribution $\mathbf{p} = (p_1, \ldots, p_T)$ of the sensitive attributes and the Gaussian error function err.

5. Experimental Evaluation

5.1 Experimental Setup

In this section, we provide experimental results of our model inversion attack for prediction systems using linear regression models given in Sect. 4. As we have mentioned in Sect. 1), linear regression models are used for resolving the cold start problem of recommendation systems [29] and plays some important roles in prediction systems. We like to think of our experimental result as a proof of concept, rather than a full-fledged analysis of our proposed attack. Our result indicates that model inversion attacks can indeed succeed without the knowledge of the non-sensitive attributes. In the following, we provide the two data sets we used to evaluate the performance of our attack.

- **How Americans Like Their Steak [30].** We refer to this data set as *data set A*. Data set A is a survey conducted by FiveThirtyEight. The survey collects the response of Americans for the question "How do you like your steak prepared", along with their demographic information such as household income, age, and gender. For our experiment, we assume a scenario where a prediction system PredSys using linear regression as the prediction model predicts the steak preference of a user based on the household income, age, gender, drinking habit, and smoking habit. Here, since the household income had five categories, we used the one-hot-encoding (i.e., this attribute was represented by a five-bit binary string). The input/output attributes are provided in Table 1. We used the data of 335 users that did not include a missing value in any attribute. The data of 200 users were used to train the prediction model $f_{\mathbf{w}}$ and the data of the remaining 135 users were used for performance evaluation.
- **MovieLens 1M Dataset [31].** We refer to this data set as *data set B*. Data set B was collected by GroupLens Research, and contains as attributes movie rating, occupation, age, gender, and timestamp. For our experiment, we assume a similar scenario as above, where instead the prediction system predicts the movie rating of a user based on the occupation, age, and gender. Since occupation has 21 categories, we used a 21-bit binary string to represent the values. The input/output attributes are provided in Table 2. We divided data set B into two data sets based on time. The former set consisting of 727,376 ratings were used to train the prediction model $f_{\mathbf{w}}$ and the latter set consisting of 272,689 ratings were used for performance evaluation.

For both data sets A and B, we divided the input attributes $\mathbf{x}$ into sensitive attributes and non-sensitive attributes. As we have discussed in Sect. 1, what the users consider as sensitive attributes and non-sensitive attributes may differ. Therefore, in our experiment, we tested several combinations of sensitive and non-sensitive attributes to evaluate the performance. Namely, we selected $(x_1, \cdots, x_T)$ as sensitive attributes for varying T, where $T = 5, 6, 7, 8$, or 9 in data set A, and $T = 21, 22$, or 23 in data set B[†].

Table 1 Attributes of data set A.

Attribute	Name	Value
x_1–x_5	Household income	Five-bit binary string ($0–$24,999, $25,000–$49,999, $50,000–$99,999, $100,000–$149,999, or larger than $150,000)
x_6	Age	0 (18–29), 1 (30–44), 2 (45–60), or 3 (61–)
x_7	Gender	0 (woman) or 1 (man)
x_8	Drinking habit	0 (not drink) or 1 (drink)
x_9	Smoking habit	0 (not smoke) or 1 (smoke)
y	Preference of steak	1 (rare), 2 (medium rare), 3 (medium), 4 (medium well), or 5 (well)

Table 2 Attributes of data set B.

Attribute	Name	Value
x_1–x_{21}	Occupation	21-bit binary string (21 categories)
x_{22}	Age	0 (under 18), 1 (18–24), 2 (25–34), 3 (35–44), 4 (45–49), 5 (50–55), or 6 (56–)
x_{23}	Gender	0 (woman) or 1 (man)
y	Rating	1, 2, 3, 4, or 5

We used a fixed value $\eta = 0.01$ as the learning rate in both data sets, and trained the prediction model $f_{\mathbf{w}}$ using the aforementioned training data. We selected random malicious data from the training data (we attempted 20 ways to randomly select malicious data), and performed data poisoning using the algorithms Setup and Poisoning provided in Sect. 4.3. After the data poisoning procedure, we performed our model inversion attack for the evaluation data. Specifically, we inferred, for each evaluation data point, the sensitive attributes from the output using the ModelInversion algorithm.

We evaluated our attack by the number of malicious data, the accuracy of the prediction model $f_{\mathbf{w}}$, and the accuracy of our model inversion attack. For the accuracy of the prediction model $f_{\mathbf{w}}$, we used RMSE on the evaluation data. For the accuracy of our model inversion attack, we evaluated the *success rate*, which we define as the ratio of the number of successful attacks (i.e., attacks in which all of the inferred sensitive attributes coincide with the true sensitive attributes) divided by the number of attacks (i.e., the size of the evaluation data). For comparison, we also evaluated the success rate of the model inversion attack proposed by Fredrikson et al. [15], [16] where the adversary is provided with all the non-sensitive attribute values prior to the attack. We took the average of the number of malicious data, RMSE, and the success rate over all the 20 ways of randomly selecting malicious data to stabilize performance.

[†]The way we chose the sensitive attributes are highly influenced on what we thought should be considered as sensitive. For example, it seemed safe to say that household income $x_1, \cdots, x_5$ is considered to be sensitive for most users.

Table 3 Experimental results (data set A).

Sensitive attributes	Non-sensitive attributes	Average number of malicious data	RMSE	Success rate (Our attack)	Success rate (Attack of Fredrikson et al.)
x_1–x_9	–	–	0.985	–	0.233
x_1–x_8	x_9	5.9	1.009	0.241	0.244
x_1–x_7	x_8–x_9	46.3	1.013	0.281	0.274
x_1–x_6	x_7–x_9	66.0	1.018	0.497	0.504
x_1–x_5	x_6–x_9	79.0	1.040	0.752	0.778

Table 4 Experimental results (data set B).

Sensitive attributes	Non-sensitive attributes	Average number of malicious data	RMSE	Success rate (Our attack)	Success rate (Attack of Fredrikson et al.)
x_1–x_{23}	–	–	0.953	–	0.313
x_1–x_{22}	x_{23}	20.0	0.961	0.359	0.367
x_1–x_{21}	x_{22}–x_{23}	71.8	0.967	0.614	0.618

5.2 Evaluation Results

Our experimental result is illustrated in Tables 3 and 4. Note that we provide the "average" number of malicious data. This is because we took the average of the number of malicious data over 20 ways of randomly selecting malicious data, as described in Sect. 5.1. Observe that the average number of malicious data was about 80 at most. Although some may think that collecting 80 malicious data is difficult for an adversary, we consider our result as an indication to a realistic and possible threat. In some scenarios where adversaries are colluding, it can be a rather easy task to collect around 100 actual data points by pooling the data from themselves and their friends. As discussed in Sect. 4.2.2, the adversaries can use these pooled data as actual training data, and substitute them as the source data set $\mathbf{z}_{\mathsf{DB}}$ for generating the malicious data set.

Tables 3 and 4 show that RMSE were hardly increased by data poisoning. More specifically, when the adversary did not perform the data poisoning, RMSE was 0.985 and 0.953 in data sets A and B, respectively. By performing the data poisoning, RMSE only increased at most 0.055 and 0.014 in data sets A and B, respectively. Since the output values were rounded to 1, 2, 3, 4, or 5 in both the data sets, this slight increase of RMSE does not have any significant affect to the output of the model in most cases. Thus, it seems safe to say that the adversary successfully modified the prediction model into a target prediction model without much degradation of the prediction accuracy.

Finally, Tables 3 and 4 indicate that the success rate of our model inversion attack was almost the same as that of the model inversion attack in [15], [16]. We emphasize again that in the model inversion attack in [15], [16], the adversary has to obtain all of the non-sensitive attributes, which may not be an easy task (as described in Sect. 1). On the other hand, our model inversion attack does not require the knowledge of the non-sensitive attributes, and achieves almost the same success rate as the model inversion attack in [15], [16].

Besides, as mentioned in Sect. 2.1, even if the adversary regarded all of the non-sensitive attributes as sensitive ones (i.e., $T = D$), the success rate of the model inversion attack in [15], [16] is very low; i.e., 0.233 and 0.313 in the data sets A and B, respectively. On the other hand, the success rate of our proposed attack is very high; e.g., 0.752 and 0.614 in the data sets A and B, respectively. Thus, when some or all of the non-sensitive attributes are unknown, data poisoning is much more effective in terms of attack accuracy than regarding the unknown non-sensitive attributes as sensitive ones.

Therefore, we can conclude that our model inversion attack successfully inferred sensitive attributes without requiring the knowledge of the non-sensitive attributes.

6. Conclusion

In this paper, we proposed a new framework called the general model inversion (GMI) framework, which models the amount of auxiliary information available to the adversary. This framework includes the model inversion attack in [15], [16] as a special case, and enables a new type of model inversion attack that does not require the knowledge of non-sensitive attributes. We then proposed an algorithm for model inversion attacks against linear regression models. The experimental results using two real data sets showed that the adversary using our model inversion attack successfully inferred sensitive attributes without non-sensitive attributes. The results also showed that our model inversion attack did not degrade the prediction accuracy of the prediction model.

Although we successfully removed non-sensitive attributes from the auxiliary information, our model inversion attack still requires the prior distribution $\mathbf{p}$ as auxiliary information (in the same way as the model inversion attack in [15], [16]). We would like to develop a model inversion attack that does not require the prior distribution $\mathbf{p}$ as future work. We would also like to develop a model inversion attack against other ML models than linear regression models, such as SVM, decision tree, and deep neural network.

References

[1] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes," Proc. 15th International Conference on Privacy, Security and Trust (PST 2017), pp.1–10, 2017.
[2] X. Su and T.M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, vol.2009, no.4, pp.1–19, 2009.
[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," IEEE Internet Comput., vol.7, no.1, pp.76–80, 2003.
[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol.42, no.8, pp.30–37, 2009.
[5] A.Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," Proc. 2013 IEEE International Conference on Data Engineering (ICDE 2013), pp.254–265, 2013.
[6] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating next-cell predictors with extensive Wi-Fi mobility data," IEEE Trans. on Mobile Comput., vol.5, no.12, pp.1633–1649, 2006.
[7] P.C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Destination prediction by trajectory distribution based model," IEEE Trans. Intell. Transport. Syst., vol.19, no.8, pp.2470–2481, 2018.
[8] The International Warfarin Pharmacogenetics Consortium, "Estimation of the Warfarin Dose with Clinical and Pharmacogenetic Data," New England Journal of Medicine, vol.360, no.8, pp.753–764, 2009.
[9] Academy of Medical Sciences, "Stratified, personalised or p4 medicine: A new direction for placing the patient at the centre of healthcare and health education," Tech. Rep., 2015.
[10] J.C. Weiss, S. Natarajan, P.L. Peissig, C.A. McCarty, and D. Page, "Machine learning for personalized medicine: Predicting primary myocardial infarction from electronic health records," AI Mag., vol.33, no.4, pp.33–45, 2012.
[11] N. Cesa-Bianchi and G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.
[12] E. Hazan, "Introduction to Online Convex Optimization," Now Publishers Inc., 2016.
[13] J.A. Calandrino, A. Kilzer, A. Narayanan, E.W. Felten, and V. Shmatikov, ""You might also like:" Privacy risks of collaborative filtering," Proc. 32nd IEEE Symposium on Security and Privacy (S&P 2011), pp.231–246, 2011.
[14] A. Friedman, B.P. Knijnenburg, K. Vanhecke, L. Martens, and S. Berkovsky, "Privacy aspects of recommender systems," in Recommender Systems Handbook, pp.649–688, Springer, 2015.
[15] M. Fredrikson, E. Lantz, and S. Jha, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," Proc. 23rd USENIX Security Symposium (USENIX 2014), pp.17–32, 2014.
[16] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015), pp.1322–1333, 2015.
[17] M. Kearnsy and M. Liz, "Learning in the presence of malicious errors," vol.22, no.4, pp.807–837, 2012.
[18] P. Auer and N. Cesa-Bianchi, "On-line learning with malicious noise and the closure algorithm," vol.23, no.1, pp.83–99, 1998.
[19] N.H. Bshouty, N. Eiron, and E. Kushilevitz, "PAC learning with nasty noise," Proc. 10th International Conference on Algorithmic Learning Theory (AAL 1999), pp.206–218, 1999.
[20] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," Proc. 29th International Conference on Machine Learning (ICML 2012), pp.1467–1474, 2012.
[21] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," IEEE Trans. Knowl. Data Eng., vol.26, no.4, pp.984–996, 2014.
[22] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), pp.99–108, 2004.
[23] D. Lowd and C. Meek, "Adversarial learning," Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005), pp.641–647, 2005.
[24] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, and J.D. Tygar, "Can machine learning be secure?," Proc. 2006 ACM Symposium on Information, computer and communications security (ASIACCS 2006), pp.16–25, 2006.
[25] L. Huang, A.D. Joseph, B. Nelson, B.I.P. Rubinstein, and J.D. Tygar, "Adversarial machine learning," Proc. 4th ACM Workshop on Artificial Intelligence and Security (AISec 2011), 2011.
[26] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," Proc. 27th International Conference on Neural Information Processing Systems, pp.2087–2095, 2014.
[27] B. Li and Y. Vorobeychik, "Scalable optimization of randomized operational decisions in adversarial classification settings," Proc. 18th International Conference on Artificial Intelligenceand Statistics, pp.599–607, 2015.
[28] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," Proc. 3rd Neural Information Processing Systems (NIPS 2016), pp.1–13, 2016.
[29] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," Expert Systems with Applications: An International Journal, vol.41, no.4, pp.2064–2073, 2014.
[30] W. Hickey, "FiveThirtyEight: How Americans Like Their Steak." http://fivethirtyeight.com/datalab/how-americans-like-their-steak/, 2014.
[31] GroupLens Research, "MovieLens 1M Dataset." http://grouplens.org/datasets/movielens/, 2003.
[32] X. Wu, M. Fredrikson, S. Jha, and J.F. Naughton, "A methodology for formalizing model-inversion attacks," Proc. 29th IEEE Computer Security Foundations Symposium (CSF 2016), pp.355–370, 2016.
[33] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," Proc. 36th IEEE Symposium on Security and Privacy (S&P 2015), pp.463–480, 2015.
[34] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2010.

Seira Hidano received his M.E. and Ph.D. degrees in computer science and engineering from Waseda University, Japan, in 2009 and 2012, respectively. In 2010, he was a JSPS research fellow. In 2011 and 2012, he was a research assistant at Waseda University. In 2013, he joined KDDI. He is currently a research engineer of the Information Security Lab. in KDDI Research, Inc. His research interest includes biometric authentication, information theoretic security, and privacy preservation.

Takao Murakami received his M.E. and Ph.D. from the University of Tokyo in 2006 and 2014, respectively. He is currently a researcher with the National Institute of Advanced Industrial Science and Technology (AIST). His research interest is biometrics and privacy. He received the DoCoMo Mobile Science Award in 2016, the IEEE TrustCom Best Paper Award in 2015, and IPSJ Yamashita SIG Research Award in 2011. He is a member of IEEE, IEICE, and IPSJ.

Shuichi Katsumata received his B.E. and M.E. in mathematical engineering and information physics from the University of Tokyo in 2016. He is a doctor course student at the University of Tokyo. He received the SCIS'16 and CSS'17 paper prize.

Shinsaku Kiyomoto received his B.E. in engineering sciences and his M.E. in Material Science from Tsukuba University, Japan, in 1998 and 2000, respectively. He joined KDD (now KDDI) and has been engaged in research on stream ciphers, cryptographic protocols, and mobile security. He is currently a senior researcher at the Information Security Laboratory of KDDI Research, Inc. He was a visiting researcher of the Information Security Group, Royal Holloway University of London from 2008 to 2009. He received his doctorate in engineering from Kyushu University in 2006. He received the IEICE Young Engineer Award in 2004 and Distinguished Contributions Awards in 2011. He is a member of IEICE and JPS.

Goichiro Hanaoka graduated from the Department of Engineering, the University of Tokyo in 1997. He received the Ph.D. degree from the University of Tokyo in 2002. He joined AIST in 2005. Currently, he is Leader of the Advanced Cryptosystems Research Group, Information Technology Research Institute, AIST. He engages in the R&D for encryption and information security technologies including the efficient design and security evaluation of public key cryptosystems. He has received numerous awards including the DoCoMo Mobile Science Award (2016), Mobile Communication Fund; the Wilkes Award (2007), British Computer Society; Best Paper Award (2008), The Institute of Electronics, Information and Communication Engineers (IEICE); and Innovative Paper Awards (2012, 2014), Symposium on Cryptography & Information Security (SCIS), IEICE.