Membership Inference Attacks against Machine Learning Models

Reza Shokri
Cornell Tech
shokri@cornell.edu

Marco Stronati
Cornell Tech
ms3252@cornell.edu

Vitaly Shmatikov
Cornell Tech
shmat@cs.cornell.edu

arXiv:1610.05820v1 [cs.CR] 18 Oct 2016

***Abstract*—We investigate how machine learning models leak information about the individual data records on which they were trained. We focus on the basic membership inference attack: given a data record and black-box access to a model, determine whether the record was in the model's training dataset. To perform membership inference against a target model, we make adversarial use of machine learning and train our own inference attack model to recognize differences in the target model's predictions on inputs that it trained on versus inputs that it did not use during training.**

We empirically evaluate our inference techniques on classification models trained by commercial "machine learning as a service" providers such as Google and Amazon. Using realistic datasets and classification tasks, we show that these models can be significantly vulnerable to membership inference attacks.

I. Introduction

Machine learning is the foundation of popular Internet services such as image and speech recognition and natural language translation. Many companies also use machine learning internally, to improve marketing and advertising, recommend products and services to users, or better understand the data generated by their operations. In all of these scenarios, activities of individual users—their purchases and preferences, health data, online and offline transactions, photos they take, commands they speak into their mobile phones, locations they travel to—are used as the training data.

Internet giants such as Google and Amazon are already offering "machine learning as a service." Any customer in possession of a dataset and a data classification task can upload this dataset to the service. The service automatically constructs a model and makes it available to the customer, typically as a black-box API. For example, a mobile-app maker can use this service to analyze users' activities and use the resulting model inside the app to promote in-app purchases to users when they are most likely to respond. Machine-learning services also let data owners expose their models to external users for querying or sell them while keeping the training data confidential.

Our contributions. We investigate what, given black-box access to a machine learning model, can be inferred about the model's training dataset. We start with the fundamental question that we call **membership inference**: given a model and a record, determine whether this record was used to train the model. This question is interesting from both the machine learning and privacy perspectives. We answer it quantitatively in the most difficult setting, where the adversary's access to the model is limited to "black-box" `predict()` calls that return the model's output on a given input.

To answer the membership inference question, we turn machine learning against itself and train an *attack model* whose purpose is to distinguish the target model's behavior on training inputs from its behavior on inputs that it did not encounter during training. In other words, we turn the membership inference problem into a classification problem.

Attacking black-box models, e.g., those built by commercial "machine learning as a service" providers, requires more sophistication than attacking white-box models whose structure and parameters are known to the adversary. To construct the attack model, we invented a new *shadow training* technique. First, we create multiple "shadow models" that imitate the behavior of the target model, but for which we know the training datasets and thus the ground truth about membership in these datasets. We then train the attack model on the inputs and outputs of the shadow models.

We developed several effective methods to generate training data for the shadow models. The first method uses black-box access to the target model to synthesize this data. The second method uses statistics about the population from which the target's training dataset was drawn. The third method assumes that the adversary has access to a noisy version of the target's training dataset. The first method does not assume any prior knowledge about the distribution of the target model's training data, while the second and third methods allow the attacker to query the target model only *once* before inferring whether a given record was in its training dataset.

Our inference techniques are generic and not based on any particular dataset or model type. We evaluate them on neural networks, as well as black-box models trained using Amazon ML and Google Prediction API. All of our experiments on Amazon and Google were done without knowing the learning algorithms used by these services, nor the architecture of the resulting models, since they don't reveal this information to the users. For our evaluation, we use concrete datasets of images, retail purchases, and location traces, realistic classification tasks, and standard model-training procedures. In addition to demonstrating that inference attacks are successful, we quantify how their success relates to the classification tasks and to the standard machine learning metrics for overfitting.

Our experimental results show that models created using machine-learning-as-a-service platforms can leak a lot of information about their training datasets. For multi-class classification models trained on 10,000-record retail purchase transaction datasets using Google and Amazon with default configurations, our membership inference achieves median accuracy of 94% and 74%, respectively. The attack accuracy against Google-trained models is 90% if shadow training is performed on fully synthetic data.

Inferring information about the model's training dataset should not be confused with procedures such as model inversion that use a model's output on a hidden input to infer something about this input [13] or extract features that characterize one of the model's classes [12]. As explained in [22] and Section II, model inversion does not produce an actual member of the model's training dataset, nor, given a record, infer whether this record was in the training dataset. By contrast, the membership inference problem we study in this paper is similar, in spirit, to the problem of identifying the presence of an individual's data in a mixed pool given some statistics about the pool [16], [24].

In summary, this paper demonstrates and quantifies the problem of machine learning models leaking information about their training datasets. To train our attack models, we developed a new shadow learning technique that works with minimal knowledge about the target model and its training dataset. Finally, we quantify how leakage of membership information is related to model overfitting.

II. Related Work

Attacks on statistical and machine learning models. In [2], knowledge of the parameters of SVM and HMM models is used to infer *general statistical information* about the training dataset, for example, whether records of a particular race were used during training. By contrast, our inference attacks work in the black-box setting, without any knowledge of the model's parameters, and infer information about *specific records* in the training dataset, as opposed to general statistics.

Homer et al. [16] developed techniques for inferring the presence of a particular genome in a dataset. Their approach involves comparing the published statistics about this dataset (in particular, minor allele frequencies) to the distribution of these statistics in the general population. By contrast, our inference attacks target trained machine learning models, not specific statistics.

Other attacks on machine learning include [5], where the adversary exploits *changes* in the outputs of a collaborative recommender system to infer inputs that caused these changes. These attacks exploit temporal behavior specific to the recommender systems based on collaborative filtering.

Model inversion. Model inversion [12], [13] uses the output of a model applied to a hidden input to infer certain features of this input. See [22] for a detailed analysis of this attack and an explanation of why it does not necessarily entail a privacy breach. For example, in the specific case of pharmacogenetics analyzed in [13], the model captures the correlation between the patient's genotype and the dosage of a certain medicine.

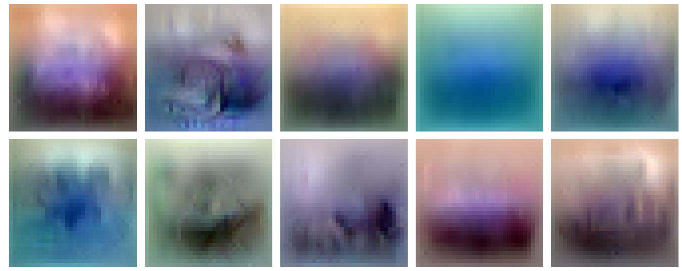

Fig. 1. Images produced by model inversion on a trained CIFAR-10 model. Top: airplane, automobile, bird, cat, deer. Bottom: dog, frog, horse, ship, truck. The images do not correspond to any specific image in the training dataset, are not human-recognizable, and at best (e.g., the truck class image) are vaguely similar to the average image of all objects in a given class. (Analysis and image courtesy of Congzheng Song.)

This correlation is a valid scientific fact that holds for all patients, regardless of whether they were included in the model's training dataset or not. It is not possible to prevent disclosure due to population statistics [11].

In general, model inversion cannot be used to determine if a particular record was used as part of the model's training dataset (i.e., membership inference). Given a record and a model, model inversion works exactly the same way when the record was used to train the model and when it was *not* used. In the case of pharmacogenetics [13], model inversion produces almost identical results for members and non-members. Due to the slight overfitting of the model, the results are a little bit (4%) more accurate for the members, but this accuracy can only measured in retrospect, if the adversary already knows the ground truth (i.e., which records are indeed members of the model's training dataset). By contrast, our goal is to construct a decision procedure that clearly distinguishes members from non-members.

Model inversion has also been applied to face recognition models [12]. In this scenario, the model's output is set to 1 for class i and 0 for the rest, and model inversion is used to construct an input that produces these outputs. This input is *not* an actual member of the training dataset, but simply an average of the features that "characterize" the class.

In the face recognition scenario—and *only* in this specific scenario—each output class of the model is associated with a single person. The training images for this class are different photos of that person. Model inversion constructs an artificial image that is somehow an average of these training images. Because they all happen to depict the same person, their average is recognizable (by a human) as that person. Critically, model inversion does not produce any *specific image* from the training dataset, which is the definition of membership inference.

In any other scenario, where the class involves multiple individuals or objects, the results of model inversion are semantically meaningless. To illustrate this, we ran model inversion against a convolutional neural network[1] trained on the CIFAR-10 dataset, which is a standard benchmark for

[1] github.com/Lasagne/Recipes/blob/master/modelzoo/cifar10_nin.py

object recognition models. Each class includes different images of a single object (e.g., an airplane). Figure 1 shows the images "reconstructed" by model inversion. As expected, they do not correspond to *any* image with a recognizable object, let alone an image from the training dataset. We expect similar results for other models, too. For example, for the pharmacogenetics model mentioned above, this form of model inversion produces the average of different patients' genomes. For the model that classifies location traces into geosocial profiles (see Section VII-A), model inversion produces the average of the location traces of different people. In both cases, the results of model inversion are not associated with any specific individual, nor any specific input from the training dataset.

In summary, model inversion produces the average of the features that characterize an output class. It does *not* (1) construct a specific member of the training dataset, nor (2) given an input and a model, determines if this specific input was used to train the model.

Model extraction. Model extraction attacks [27] aim to extract the parameters of a model trained on private training data. The attacker's goal is to construct a model whose predictive performance on validation data is similar to target model.

Model extraction can be a stepping stone for inferring information about the model's training dataset. In [27], this is illustrated for a specific type of models called kernel logistic regression (KLR) [34]. In KLR models, the kernel function includes a tiny fraction of the training data (so called "import points") directly into the model. Since import points are parameters of the model, extracting them results in the leakage of that particular part of the data. This result is very specific to KLR and does not generalize to other types of models since they do not explicitly store training data in their parameters.

Even in the case of KLR models, leakage is not quantified, other than via visual similarity of a few chosen import points and "the closest (in L1 norm) extracted representers." The results are shown only for the MNIST handwritten digit dataset, where all members of a class are very similar (e.g., all members of the first class are different ways of writing digit 1). Thus, any extracted digit for a class would be similar to all images associated with that class, whether they were in the training set or not.

In summary, [27] does not provide a general method of solving the membership inference problem, that is, given an input and a model, determine whether this specific input was used to train the model.

Privacy-preserving machine learning. Existing literature on privacy protection in machine learning focuses mostly on how to learn without direct access to the training data. Secure multiparty computation (SMC) has been used for learning decision trees [21], linear regression functions [9], association rules [28], Naive Bayes classifiers [29], and k-means clustering [17]. The goal is to limit information leakage during training. The training algorithm is the same as in the non-privacy-preserving case, thus the resulting models are as vulnerable to inference attacks as any conventionally trained model. This also holds for models trained by computing on encrypted data [3], [4], [31].

Differential privacy [10] has been applied to linear and logistic regression [6], [33], support vector machines [23], risk minimization [7], [30], deep learning [1], [25], learning an unknown probability distribution over a discrete population from random samples [8], and releasing hyper-parameters and classifier accuracy [20]. In [25], multiple parties jointly train a deep neural-network model while sharing differentially private changes in their respective model parameters. By definition, differentially private training produces models that do not depend too much on any particular member of the training dataset. We leave the design of differentially private training procedures that prevent our inference attacks to future work.

III. MACHINE LEARNING BACKGROUND

Machine learning algorithms help us better understand and analyze complex data. When the model is created using *unsupervised* training, the objective is to extract useful features from unlabeled data and build a model that explains its hidden structure. When the model is created using *supervised* training, which is the focus of this paper, the training records (as inputs of the model) are assigned labels or scores (as outputs of the model). The goal is to learn the relationship between the data and the labels and construct a model that can generalize to data records beyond the training set [15]. Model-training algorithms aim to minimize the model's prediction error on the training dataset and thus may overfit to this dataset, producing models that perform better on the training inputs than on inputs drawn from the same population but not used during the training. Many *regularization* techniques have been proposed to prevent models from becoming too dependent on their training datasets while minimizing their prediction error [15].

Supervised training is often used for classification and other prediction tasks. For example, a retailer may train a model that predicts a customer's shopping style to offer her suitable incentives, while a medical researcher may train a model to predict which treatment is most likely to succeed given a patient's clinical symptoms or genetic makeup.

Machine learning as a service. Major Internet companies now offer machine learning as a service in their cloud platforms. Examples include Google Prediction API,[2] Amazon Machine Learning (Amazon ML),[3] Microsoft Azure Machine Learning (Azure ML),[4] and BigML.[5]

These platforms provide simple APIs for uploading the data and training and querying models, thus making state-of-the-art machine learning technologies available to any developer. For example, a developer may create an app that gathers data from users, uploads it into the cloud platform to train a model (or update an existing model with new data), and

[2]cloud.google.com/prediction
[3]aws.amazon.com/machine-learning
[4]studio.azureml.net
[5]bigml.com

then uses the model's predictions inside the app to improve its features or better interact with the users. Some platforms even envision data holders training a model and sharing it through the platform's API for profit.[6]

The details of the models and the training are hidden from the data owners. The type of the model may be chosen by the service adaptively, depending on the data and perhaps accuracy on validation subsets. Service providers do not warn users about the consequences of overfitting and provide little or no control over regularization. For example, Google Prediction API hides all details, while Amazon ML provides only a very limited set of pre-defined options (L1- or L2-norm regularization). The models cannot be downloaded and are accessed only through the service's API. Service providers derive revenue mainly by charging developers for queries through this API. Therefore, for the rest of this paper, we treat "machine learning as a service" as a black box. All inference attacks we demonstrate are performed entirely through the services' standard APIs.

IV. Privacy in Machine Learning

A. Inference about members of the population

Before dealing with inference attacks, we need to define what privacy means in the context of machine learning or, alternatively, what it means for a machine learning model to breach privacy. A plausible notion of privacy, known as the "Dalenius desideratum," states that the model should reveal no more about the input to which it is applied than would have been known about this input without applying the model. This cannot be achieved by any useful model [11].

A related notion of privacy appears in prior work on model inversion [13]: a privacy breach occurs if an adversary can use the model's output to infer the values of unintended (sensitive) attributes used as input to the model. As observed in [22], it may not be possible to prevent this "breach" if the model is based on scientific or statistical facts about the population. For example, suppose that training the model has uncovered a high correlation between a person's externally observable phenotype features and genetic predisposition to a certain disease. This correlation is now a publicly known scientific fact, which allows anyone to infer information about the person's genome after observing that person.

Critically, this correlation is true for *all* members of a given population. Therefore, the model breaches privacy not just of people whose data was used to create the model, but also of every other person in the population, even those whose data was not used and whose identities are not known to the model creator (i.e., this is "spooky action at a distance"). There is nothing the model creator can do to protect their privacy, since the correlation underlying the model remains true regardless of the model creator's actions.

B. Inference about members of the training dataset

To bypass the difficulties inherent in defining and protecting privacy of the general population, we focus on a narrower, more manageable goal: protecting privacy of the individuals whose data was used to train the model. This motivation is closely related to the original goals of differential privacy.

Of course, members of the training dataset are members of the population, too. We investigate what the model reveals about them *beyond* what it reveals about an arbitrary member of the population. Our ultimate goal is to measure the specific *membership risk* that a person incurs if they allow their data to be used to train a model.

The basic inference attack is **membership inference**, i.e., determining whether a given data record was part of the model's training dataset. When a record is fully known to the adversary, learning that it was used to train a particular model is an indication of information leakage through the model, and in some cases can directly lead to a privacy breach. For example, knowledge that a certain patient's record was used to train a model associated with a disease (e.g, to determine appropriate medicine dosage or discover the genetic basis of the disease) can reveal that the patient has this disease.

We investigate this problem in the **black-box** setting where the adversary can only supply inputs to the model and receive the model's output(s). In some situations, the model is available to the adversary indirectly. For example, an app developer may use a machine-learning service to construct a model from the data collected by the app and have the app make API calls to the resulting model. In this case, the adversary would supply inputs to the app (rather than directly to the model) and receive the app's outputs (which are based on the model's outputs). The details of internal model usage vary significantly from app to app. For simplicity and generality, we will assume that the adversary directly supplies inputs to and receives outputs from the black-box model.

V. Problem Statement

Consider a set of labeled data records sampled from some population and partitioned into classes. We assume that a machine learning algorithm is used to train a classification model for finding the relationship between the content of the data records and their labels. For any input data record, the model outputs the *prediction vector* of probabilities, one per output class, that the record belongs to that class. The class with the highest probability is selected as the predicted label for the data record. The accuracy of the model is evaluated by measuring how it generalizes beyond its training set and predicts the labels of other data records in the same population.

In general, we want to measure how much the model's output (i.e., the prediction vector) leaks about the individual data records that were used to train the model. The focus of this paper is a more specific problem: measuring how much the model's output leaks about the membership of a specific data record in the model's training dataset.

We assume that the attacker has black-box access to the model and can obtain the model's prediction vector on any data record. The attacker knows the format of the inputs and outputs of the model, including their number and the range of values they can take. We also assume that the attacker either

[6] cloud.google.com/prediction/docs/gallery

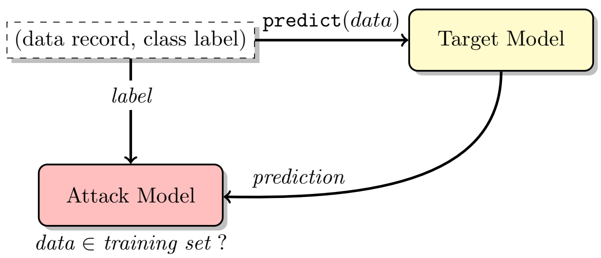

Fig. 2. Membership inference attack in the black-box setting. The attacker queries the target model with a data record and obtains the model's prediction on that record. The prediction is a vector of probabilities, one per class, that the record belongs to a given class. This prediction vector, along with the label of the target record, is passed to the attack model, which infers whether the record was *in* or *out* of the target model's training dataset.

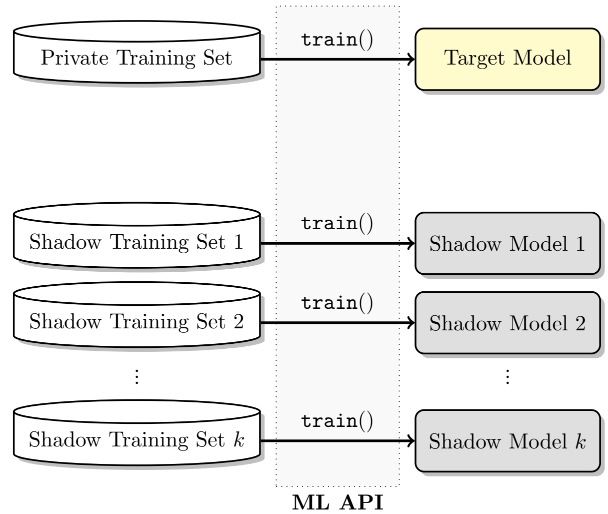

Fig. 3. Training shadow models using the same machine learning platform as was used to train the target model. The training datasets of the target and shadow models are disjoint, but the training datasets of shadow models might overlap. The formats of the target and shadow models' respective training datasets are the same, but the models' internal parameters are independent.

(1) knows the type and architecture of the machine learning model, as well as the training algorithm, or (2) has black-box access to a machine learning oracle (e.g., "machine learning as a service" platform) that has been used to train the model. In the latter case, the attacker does *not* know a priori the model's structure or meta-parameters.

The attacker may have some background knowledge about the population from which the target model's training dataset was drawn. For example, he may have independently drawn samples from the population, disjoint from the target model's training dataset. Alternatively, the attacker may know some general statistics about the population, for example, the marginal distribution of feature values.

The setting for our inference attack is as follows. The attacker is given a data record and black-box access to the target model. The attack succeeds if the attacker can correctly determine whether this data record was part of the model's training dataset or not. The accuracy of the attack can be measured via precision (what fraction of records inferred as members are truly members of the training dataset), and recall (what fraction of the training dataset's members can be correctly inferred as members by the attacker).

VI. Membership Inference

A. Overview of the attack

Our membership inference attack exploits the observation that machine learning models often behave differently on the data that they trained on versus the data they "see" for the first time. Overfitting is a common reason but not the only one (see Section VIII). The objective of the attacker is to construct an *attack model* that can recognize such differences in the target model's behavior and use them to distinguish members from non-members of the target model's training dataset based solely on the target model's output.

Our attack model is a collection of models, one for each output class of the target model. This increases accuracy because the target model produces different distributions over output classes depending on the input's true class.

To train our attack model, we build multiple "shadow" models intended to behave similarly to the target model. In contrast to the target model, we know the ground truth for each shadow model, i.e., whether a given record was in its training dataset or not. Therefore, we can use the inputs and outputs of the shadow models to perform supervised training of the attack model and thus teach it how to distinguish the shadow models' outputs on members of their training datasets from their outputs on non-members.

Formally, let $f_{\mathsf{target}}()$ be the target model, and let $D^{\mathsf{train}}_{\mathsf{target}}$ be its private training dataset which contains labeled data records $(\mathbf{x}^{\{i\}}, y^{\{i\}})_{\mathsf{target}}$. A data record $\mathbf{x}^{\{i\}}_{\mathsf{target}}$ is the input to the model, and $y^{\{i\}}_{\mathsf{target}}$ is the true label that can take values from a set of classes of size c_{target}. The output of the target model is a probability vector of size c_{target} (the values are in $[0, 1]$ and sum up to 1).

Let $f_{\mathsf{attack}}()$ be the attack model. Its input $\mathbf{x}_{\mathsf{attack}}$ is composed of the prediction vector of size c_{target} and the true label of the record. Since the goal of the attack is decisional membership inference, the inference model is a binary classifier with 2 classes corresponding to "in" and "out" labels.

Figure 2 illustrates our end-to-end attack process. For a record $(\mathbf{x}, y)$, we use the target model to compute the prediction $\mathbf{y} = f_{\mathsf{target}}(\mathbf{x})$. The distribution of $\mathbf{y}$ (classification confidence values) depends heavily on the true class of $\mathbf{x}$. This is why we pass the true label y of $\mathbf{x}$ in addition to the model prediction $\mathbf{y}$ to the attack model. Given how probabilities in $\mathbf{y}$ are distributed around y, the attack model computes the membership probability $\Pr\{(\mathbf{x}, y) \in D^{\mathsf{train}}_{\mathsf{target}}\}$.

The main challenge is how to train the attack model to distinguish members from non-members of the target model's training dataset in the black-box setting, where the attacker has no information about the internal parameters of the target model and only limited access to it through the public API. To solve this conundrum, we developed a new **shadow training** technique that lets us train the attack model on proxy targets for which we do know the training dataset and can thus perform supervised training.

B. Shadow models

The attacker creates k shadow models $f^{i}_{\mathsf{shadow}}()$. Each shadow model i is trained on a dataset $D^{\mathsf{train}}_{\mathsf{shadow}^{i}}$ that is of similar format and distributed similarly to the target's training dataset. These shadow training datasets can be generated using any of methods described in Section VI-C. Of course, we assume that the datasets used for training the attacker's shadow models do not overlap with the private dataset used to train the target model ($\forall i, D^{\mathsf{train}}_{\mathsf{shadow}^{i}} \cap D^{\mathsf{train}}_{\mathsf{target}} = \emptyset$).

The shadow models must be trained in a similar way to the target model. This is easy if the attacker already knows which machine learning algorithm (e.g., neural networks, SVM, logistic regression) was used to train the target model and has some information about the model's structure (e.g., the wiring of a neural network). Machine learning as a service is more challenging. Here the type and structure of the target model are not known, but the attacker can use exactly the same service (e.g., Google Prediction API) to train the shadow model as was used to train the target model—see Figure 3. Our empirical results confirm that this is an effective way to construct shadow models.

The larger the number of shadow models, the more accurate the attack model will be. As described in Section VI-D, the attack model is trained to recognize differences in shadow models' behavior when these models operate on inputs from their own training datasets versus inputs they did not encounter during training. Therefore, the more shadow models, the more training material for the attack model.

C. Generating training data for shadow models

To train shadow models, the attacker needs training datasets that are distributed similarly to the target model's training dataset. We developed several methods for generating such training datasets.

Model-based synthesis. If the attacker does not have access to real training data and does not know any statistics about its distribution, he can generate synthetic training data for shadow models using the target model itself. The intuition is that data records that are classified by the target model with high confidence should be statistically similar to the target model's training dataset and thus provide good fodder for shadow models.

The synthesis process runs in two phases: (1) *search*, using a hill-climbing algorithm, the space of possible data records to find records that are classified by the target model with high confidence; (2) *sample* synthetic data from these records. After this process synthesizes a record, the attacker can repeat it again and again until the training dataset for shadow models is full.

Algorithm 1 Data synthesis using the target model

procedure SYNTHESIZE(class : c)
    $\mathbf{x} \leftarrow$ RANDRECORD( ) ▷ *initilize a record randomly*
    $y^*_c \leftarrow 0$
    $j \leftarrow 0$
    $k \leftarrow k_{max}$
    for $iteration = 1 \cdots iter_{max}$ **do**
        $\mathbf{y} \leftarrow f_{\mathsf{target}}(\mathbf{x})$ ▷ *query the target model*
        if $y_c \geq y^*_c$ **then** ▷ *accept the record*
            if $y_c > \mathrm{conf}_{min}$ and $c = \arg\max(\mathbf{y})$ **then**
                if $\mathrm{rand}() < y_c$ **then** ▷ *sample*
                    return $\mathbf{x}$ ▷ *synthetic data*
                end if
            end if
            $\mathbf{x}^* \leftarrow \mathbf{x}$
            $y^*_c \leftarrow y_c$
            $j \leftarrow 0$
        else
            $j \leftarrow j + 1$
            if $j > rej_{max}$ **then** ▷ *many consecutive rejects*
                $k \leftarrow \max(k_{min}, \lceil k/2 \rceil)$
                $j \leftarrow 0$
            end if
        end if
        $\mathbf{x} \leftarrow$ RANDRECORD($\mathbf{x}^*$, k) ▷ *randomize k features*
    end for
    return $\perp$ ▷ *failed to synthesize*
end procedure

See Algorithm 1 for the pseudocode of our synthesis procedure. First, fix class c for which the attacker wants to generate synthetic data. The first phase is an iterative process. Start by randomly initializing a data record $\mathbf{x}$. Assuming that the attacker only knows the syntax of data records, sample the value for each feature uniformly at random from among all possible values of that feature. In each iteration, propose a new record. A proposed record is *accepted* only if it increases the hill-climbing objective: the probability of being classified by the target model as class c.

Each iteration involves proposing a new candidate record by changing k randomly selected features of the latest accepted record $\mathbf{x}^*$. We initialize k to k_{max}, and divide it by 2 when rej_{max} subsequent proposals are rejected. This controls the diameter of search around the accepted record to propose a new record. We set the minimum value of k to k_{min}. This controls the speed of search for new records with potentially higher classification probability y_c.

The second, sampling phase starts when the target model's probability y_c that the proposed data record is classified as belonging to class c is larger than the probabilities for all other classes and also larger than a threshold conf_{min}. This ensures that the predicted label for the record is c, and the target model is sufficiently confident in its label prediction. We

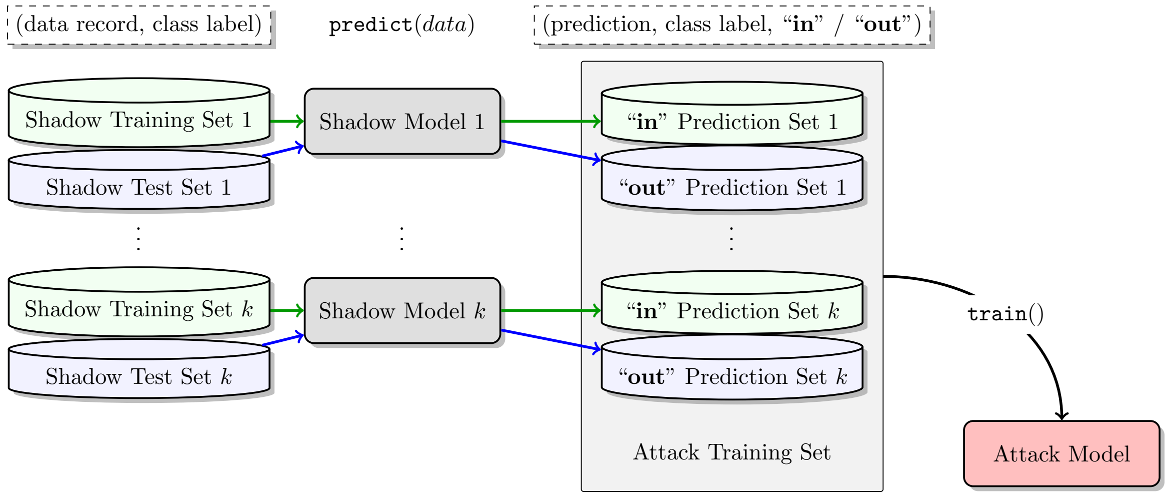

Fig. 4. Training the attack model on the inputs and outputs of shadow models. For all records in the training dataset of a shadow model, we query the model and obtain the output. These output vectors are labeled as "in" and added to the attack model's training dataset. We also query the shadow model with a test dataset disjoint from its training dataset. The outputs on this set are labeled as "out" and also added to the attack model's training dataset. Having constructed a dataset that reflects the black-box behavior of shadow models on their training and test datasets, we train a collection of c_{target} attack models, one per each output class of the target model.

select such record for the synthetic dataset with probability y_c and, if selection fails, repeat until a record is selected.

Statistics-based synthesis. The attacker may have some statistical information about the population from which the target model's training data is drawn. For example, the attacker may have prior knowledge of the marginal distributions of different features. In our experiments, we generate synthetic training records for shadow models by sampling the value of each feature independently from its own marginal distribution. The resulting attack models are very effective.

Noisy real data. Finally, the attacker may have access to noisy data that is similar to the target model's training data. In our experiments with location datasets, we simulate this by flipping the (binary) values of 10% or 20% randomly selected features, and training the shadow models on the resulting noisy dataset. This scenario models the case where the training data for the target and shadow models are not sampled from exactly the same population, or else sampled in a non-uniform way.

D. Training the attack model

The main idea behind our shadow training technique is that similar models trained on relatively similar data records using the same service behave in a similar way. This observation is empirically true and borne out by our experiments in the rest of this paper. Our results show that learning how to infer membership in shadow models' training datasets (for which we know the ground truth and can easily compute the cost function during supervised training) produces an attack model that can successfully infer membership in the target model's training dataset, too.

We query each shadow model with its own training dataset and with a disjoint test set of the same size. The outputs on the training dataset are labeled "in," the rest are labeled "out." Now, the attacker has a dataset of records, the corresponding outputs of the shadow models, and in/out labels. The objective of the attack model is to infer the labels from the records and corresponding outputs.

Figure 4 illustrates the process of training the attack model. $\forall(\mathbf{x}, y) \in D^{\text{train}}_{\text{shadow}^i}$, we compute the prediction vector $\mathbf{y} = f^i_{\text{shadow}}(\mathbf{x})$ and add the record $(y, \mathbf{y}, \text{in})$ to the attack training set $D^{\text{train}}_{\text{attack}}$. Let $D^{\text{test}}_{\text{shadow}^i}$ be a set of data records disjoint from the training set of the ith shadow model. Then, $\forall(\mathbf{x}, y) \in D^{\text{test}}_{\text{shadow}^i}$ we compute the prediction vector $\mathbf{y} = f^i_{\text{shadow}}(\mathbf{x})$ and add the record $(y, \mathbf{y}, \text{out})$ to the attack training set $D^{\text{train}}_{\text{attack}}$. Finally, we split $D^{\text{train}}_{\text{attack}}$ into c_{target} partitions, each associated with a different class label. For each class label y, we train a separate model that, given $\mathbf{y}$, predicts the in or out membership status for $\mathbf{x}$.

In effect, we convert the problem of recognizing the complex relationship between members of the training dataset and the model's output into a binary classification problem. Binary classification is a standard machine learning task, thus we can use any state-of-the-art machine learning framework or service to build the attack model. Our approach is independent of the specific method used for attack model training. For example, in Section VII we construct the attack model using neural networks and also using the same black-box Google Prediction API that we are attacking, in which case we have no control over the model structure, model parameters, or training meta-parameters—but still obtain a working attack model.

VII. Evaluation

A. Data

We evaluated our attacks on five datasets and several classification tasks.

CIFAR. CIFAR-10 and CIFAR-100 are benchmark datasets used to evaluate image recognition algorithms [19]. CIFAR-10 is composed of 32×32 color images in 10 classes, with $6,000$ images per class. In total, there are $50,000$ training images and $10,000$ test images. CIFAR-100 has the same format as CIFAR-10, but it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. We use different fractions of this dataset in our attack experiments, to show the effect of the training dataset size on the performance of the attack.

Purchases. The purchase dataset is based on Kaggle's "acquire valued shoppers" challenge dataset that contains shopping histories for several thousand individuals.[7] The purpose of the challenge is to design accurate coupon promotion strategies. Each user record contains his or her transactions over a year. The transactions include many fields such as product name, store chain, quantity, and date of purchase. For our experiments, we derived a simplified purchase dataset, where each record consists of 600 binary features. Each feature corresponds to a product and represents whether the user has purchased it or not. We construct our classification tasks by clustering the records into multiple classes, each representing a different purchase style. In our experiments, we design 5 different classification tasks with different number of classes $\{2, 10, 20, 50, 100\}$. The classification task is to predict the purchase style of a user given the 600-feature vector. We use $10,000$ randomly selected records from the purchase dataset to train the target model. The rest of the dataset contributes to the test set and (if necessary) the training sets of the shadow models.

Locations. We created a location dataset from the publicly available set of mobile users' location "check-ins" in the Foursquare social network, restricted to the Bangkok area and collected from April 2012 to September 2013 [32].[8] The check-in dataset contains $11,592$ users and $119,744$ locations, for a total of $3,298$ check-ins. We filtered out users with fewer than 25 check-ins and venues with fewer than 100 visits, which left us with $5,010$ user profiles. For each location venue, we have the geographical position as well as its location type (e.g., Indian restaurant, fast food, etc.). The total number of location types is 128. We partition the Bangkok map into areas of size $0.5km \times 0.5km$, yielding 318 regions for which we have at least one user check-in.

Each record in the resulting dataset has 446 binary features, representing whether the user visited a certain region or location type, i.e., the semantic and geographical profiles of the user. The classification task is similar to the purchase dataset. We cluster the location dataset into 30 classes, each representing a different geosocial type. The classification task is to find the user's geosocial type given her record. We use $1,600$ randomly selected records to train the target model. The rest of the dataset contributes to the test set and the training sets of the shadow models.

MNIST. MNIST is a dataset of handwritten digits formatted as 32×32 images, normalized so that the digits are located at the center of the image. The dataset is composed of $70,000$ examples [9]. We use $10,000$ randomly selected records to train the target model.

Adult (Census Income). The UCI Adult dataset includes $48,842$ records with 14 different attributes such as age, gender, education, marital status, occupation, working hours, native country, etc. The (binary) classification task is to predict whether a person makes over 50K a year based on the census attributes.[10] We use $10,000$ randomly selected records to train the target model.

B. Target models

We evaluated our inference attacks on three types of target models: two constructed by cloud-based "machine learning as a service" platforms and one we implemented locally. In all cases, our attacks treat the models as black boxes. For the cloud services, we do not even know the type or structure of the models they create. We also do not know the values of the hyper-parameters used during the training process.

Machine learning as a service. The first cloud-based machine learning service in our study is Google Prediction API. With this service, the user uploads the dataset and obtains an API for querying the resulting model; there are no configuration parameters that can be changed by the user.

The other cloud service is Amazon ML. The user cannot choose the type of the model but can control a few meta-parameters. In our experiments, we varied the *maximum number of passes* over the training data and L2 *regularization amount*. The former determines the number of training epochs and controls the convergence of model training; its default value is 10. The latter tunes how much regularization is performed on the model parameters in order to avoid overfitting. We used the platform in two configurations: the default setting (10, $1e-6$) and (100, $1e-4$).

Neural networks. Neural networks have become a very popular approach to large-scale machine learning. We use Torch7 and its nn packages,[11] a popular deep-learning library that has been used and extended by major Internet companies such as Facebook.[12]

On CIFAR datasets, we train a standard convolutional neural network (CNN) with two convolution and max pooling layers plus a fully connected layer of size 128 and a `SoftMax` layer. We use `Tanh` as the activation function. We set the learning

[7] kaggle.com/c/acquire-valued-shoppers-challenge/data

[8] sites.google.com/site/yangdingqi/home/foursquare-dataset

[9] yann.lecun.com/exdb/mnist

[10] archive.ics.uci.edu/ml/datasets/Adult

[11] github.com/torch/nn

[12] github.com/facebook/fblualib

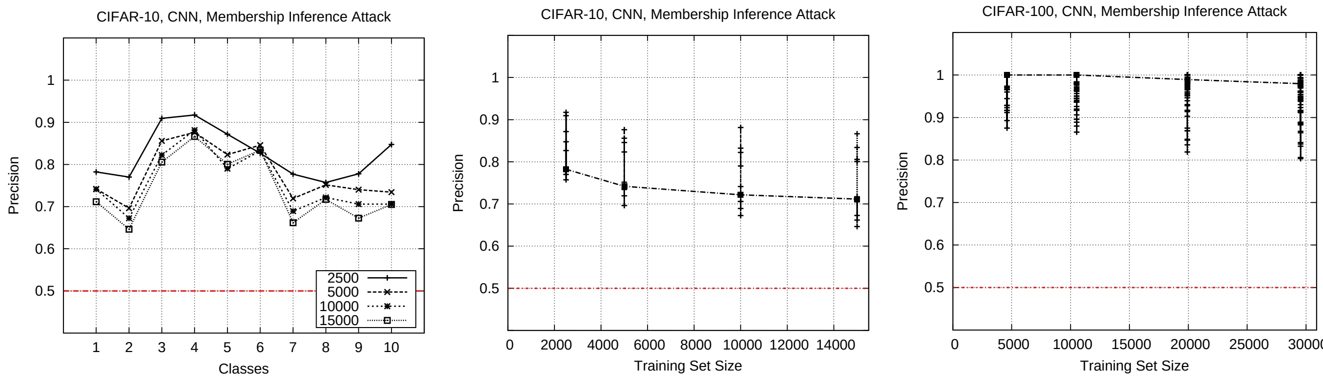

Fig. 5. The accuracy of membership inference attack against neural networks trained on CIFAR datasets. The graphs show precision accuracy for different classes when varying the size of the training datasets. The median values are connected across different training set sizes. The median precision (from the smallest dataset size to largest) is $0.78, 0.74, 0.72, 0.71$ for CIFAR-10 and $1, 1, 0.98, 0.97$ for CIFAR-100. The recall is almost 1 for both datasets. The figure on the left shows the difference in detail (for CIFAR-10). The accuracy of random guessing is 0.5.

rate to 0.001, the learning rate decay to $1e-07$, and the maximum epochs of training to 100.

On the purchase dataset (see Section VII-A), we train a fully connected neural network with one hidden layer of size 128 and a SoftMax layer. We use `Tanh` function as the activation function. We set the learning rate to 0.001, the learning rate decay to $1e-07$, and the maximum epochs of training to 200.

C. Experimental setup

The training set and the test set of each target and shadow model in our experiments are randomly selected from our datasets and have the same size. The training and test sets of each model are disjoint. There is no overlap between the datasets of the target model and those of the shadow models, but the datasets used for different shadow models can overlap with each other.

We set the training set size to $10,000$ for the purchase dataset as well as the Adult dataset and the MNIST dataset, and to $1,200$ for the location dataset. We vary the size of the training set for the CIFAR datasets, to measure the difference in the attack's accuracy. For the CIFAR-10 dataset, we choose $2,500$, $5,000$, $10,000$, and $15,000$. For the CIFAR-100 dataset, we choose $4,600$, $10,520$, $19,920$, and $29,540$.

The experiments on the CIFAR datasets were run locally, against our own models, so we can vary the model's configuration and measure the impact on the attack's accuracy. The experiments on other datasets (purchases with $\{2, 10, 20, 50, 100\}$ classes, locations, Adult, and MNIST) were run against models trained using either Google or Amazon services, where we have zero visibility into their choice of model type and structure and little control over the training process (see Section VII-B).

The main part of our evaluation focus on the purchase and location datasets. For the purchase dataset, we built target models on all platforms (Google, Amazon, local neural networks) employing the same training dataset, thus enabling us to compare the leakage from different models. We used similar training architectures for the attack models across different platforms: either a fully connected neural network with one hidden layer of size 64 with ReLU (rectifier linear units) activation functions and a SoftMax layer, or a Google-trained black-box model.

We set the number of shadow models to 100 for the CIFAR datasets, 20 for the purchase dataset, 60 for the location dataset, 50 for the MNIST dataset, and 20 for the Adult dataset. Increasing the number of shadow models would increase the accuracy of the attack but also its cost.

D. Accuracy of the attack

The attacker's goal is to determine whether a given record was part of the target model's training dataset. We evaluate this attack by executing it on (randomly reshuffled) records from the target's training and test datasets. In our attack evaluation, we use sets of the same size (i.e, equal number of members and non-members) in order to maximize the uncertainty of inference, thus the baseline accuracy is 50%.

We measure the accuracy of the attack using the standard *precision* and *recall* metrics. Precision measures the fraction of the records inferred as members of the training dataset that are indeed members. Recall measures coverage of the attack, i.e. the fraction of the training records that the attacker can correctly infer as members. Most measurements are reported per class because the accuracy of the attack can vary considerably for different classes. This is due to the difference in size and composition among classes and highly depends on the dataset.

The test accuracy of our target neural-network models with the largest training datasets (with $15,000$ and $29,540$ records, respectively) is 0.6 and 0.2 for CIFAR-10 and CIFAR-100, respectively. The accuracy is low, indicating that the models are heavily overfitted on their training sets. Figure 5 shows the results of membership inference attack against the CIFAR datasets. For both CIFAR-10 and CIFAR-100, the attack performs much better than the baseline, with CIFAR-100 especially vulnerable.

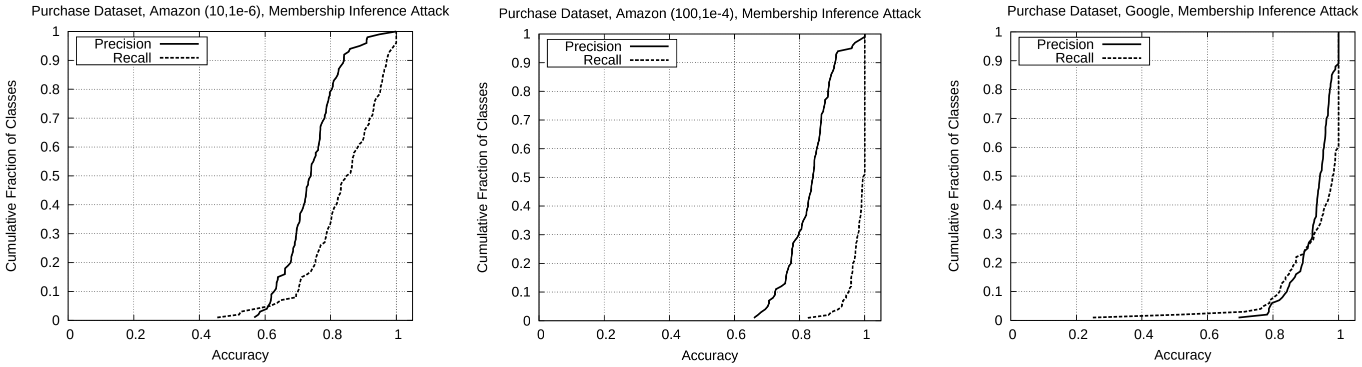

Fig. 6. Empirical CDF of precision and recall of membership inference against different classes in models trained using Amazon ML (in two different configurations) and Google Prediction API on $10,000$ purchase records. $50, 75, 90$-percentile of precision is $0.74, 0.79, 0.84$ on Amazon $(10, 1e-6)$, $0.84, 0.88, 0.91$ on Amazon $(100, 1e-4)$, and $0.94, 0.97, 1$ on Google, respectively. Recall is close to 1.

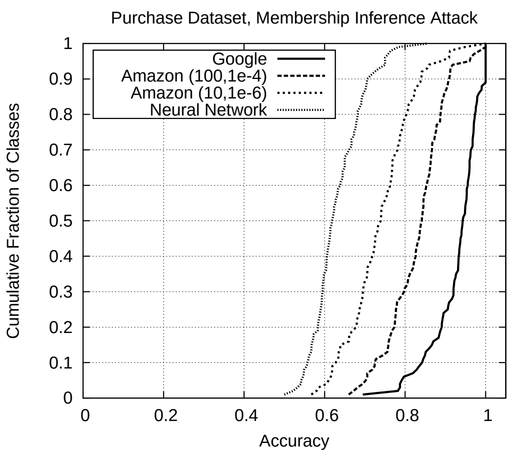

Fig. 7. Comparison between the precision accuracy of the membership inference attack on models trained on the same training datasets but using different platforms. The attack model is a neural network.

ML Platform	*Training*	*Test*
Google	0.999	0.656
Amazon (10,1e-6)	0.941	0.468
Amazon (100,1e-4)	1.00	0.504
Neural network	0.830	0.670

TABLE I
ACCURACY OF DIFFERENT ML PLATFORMS ON PURCHASE DATASET (100 CLASSES).

Table I shows the training and test accuracy of models constructed using different machine learning platforms for the purchase dataset with 100 classes. Larger gap between training and test accuracy indicates overfitting. Larger test accuracy indicates better generalizability and higher predictive power. Figure 6 shows the results of the membership inference attack against black-box models trained by Google's and Amazon's "machine learning as a service" platforms. Figure 7 compares the precision of attacks against these models with attacks against a neural-network model trained on the same data. Google's Prediction API exhibits the biggest leakage.

For the location dataset, we evaluated our attacks on Google-trained models. The training accuracy of the resulting target model is 1 and its test accuracy is 0.66. Figure 8 shows the accuracy of membership inference. Precision is between 60% and 80%, with an almost constant recall of 100%.

E. Effect of the shadow training data

Figure 8 reports the precision of attacks trained on the shadow models whose training datasets are noisy versions of some real data (disjoint from the target model's training dataset but sampled from the same population). Precision drops as the amount of noise increases, but the attack still outperforms the baseline and, even with 10% of the features in the shadows' training data replaced by random values, matches the original attack. This demonstrates that **our attacks are robust even if the attacker's assumptions about the distribution of the target model's training data are not very accurate**.

Figure 9 illustrates the accuracy of the membership inference attack when the attacker has no real data (not even noisy) for training his shadow models. Instead, we used the marginal probabilities of individual features to generate $187,300$ synthetic purchase records, then trained 20 shadow models on these records.

We also generated $30,000$ synthetic records using the model-based approach presented in Algorithm 1. In our experiments with the purchase dataset where records have 600 binary features, we initialize k to $k_{max} = 128$ and divide it by 2 when $rej_{max} = 10$ subsequent proposals are rejected. We set its minimum value $k_{min} = 4$. In the sampling phase, we set the minimum confidence threshold $conf_{min}$ to 0.2.

For our final set of sampled records, the target model's confidence in classifying the records is 0.24 on average (just a bit over our threshold $conf_{min} = 0.2$). On average, each synthetic record needed 156 queries (of proposed records) during our hill-climbing two-phase process (see Section VI-C). We trained 8 shadow models on this data.

Figure 9 compares the precision of the attack model when shadow models are trained on real data versus shadow models that are trained on synthetic data. The attack's overall precision

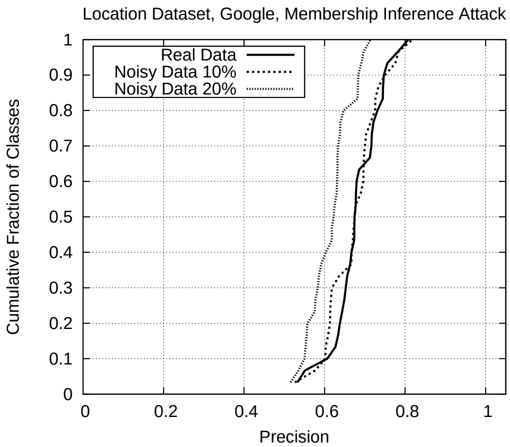

Fig. 8. Empirical CDF of the precision of membership inference attack on Google-trained models for the location dataset. Results are shown for shadow models trained on real data as well as **noisy data** with 10% and 20% noise (i.e., $x\%$ of features are replaced with random values). The precision of the attack over all classes is 0.678 (real data), 0.666 (data with 10% noise), and 0.613 (data with 20% noise). The corresponding overall recall of the attack is 0.98, 0.99, and 1.00, respectively.

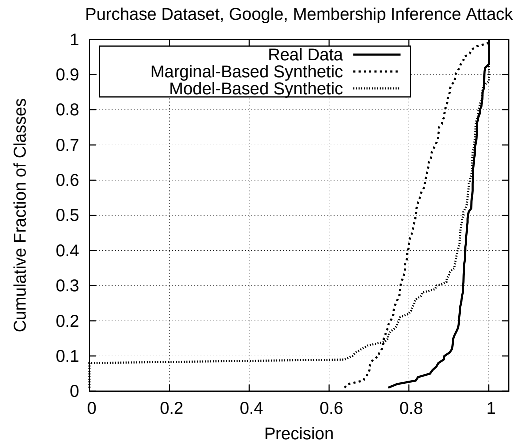

Fig. 9. Empirical CDF of the precision of membership inference attack on Google-trained model for the purchase dataset. Results are shown for different ways of generating training data for shadow models (real, synthetic generated from the target model, synthetic generated from the marginal statistics). The precision of the attack over all classes is 0.935 (real data), 0.795 (marginal-based synthetic data), and 0.896 (model-based synthetic data). The corresponding overall recall of the attack is 0.994, 0.991, and 0.526, respectively.

is 0.935 on real data compared to 0.795 for the marginal-based synthetics and 0.895 for the model-based synthetics. The accuracy of the attack using marginal-based synthetic data is noticeably reduced versus real data, but is nevertheless very high for most classes. The attack using model-based synthetic data exhibits dual behavior. For most classes its precision is high and close to the attacks using real data, but for a few classes precision is very low (less than 0.1).

The reason is that the classifier cannot confidently represent the distribution of data records in these classes (as it has not seen enough examples). The same classes are under-represented in the target model's training dataset. For example, each of the classes where the attack has less than 0.1 precision contributes under 0.6% of the target model's training dataset. Some of these classes have fewer than 30 records (out of $10,000$) in the training dataset. This makes it very difficult for the synthesis algorithm to find meaningful records for these classes while searching the high-dimensional space of possible records.

For the majority of classes, our attack achieves high precision. This demonstrates that **a membership inference attack can be trained with only black-box access to the target model, without any prior knowledge or assumptions about the distribution of the target model's training data.**

F. Effect of overfitting

The more overfitted a model, the more leakage one would expect—but only for models of the same type. For example, compare Amazon ML models trained with different configurations. The Amazon $(100, 1e-4)$ model that, according to Table I, is more overfitted leaks more than the Amazon $(10, 1e-6)$ model. However, they both leak less than the Google-trained model, even though the Google model is *less* overfitted than one of the Amazon models and has a much better predictive power (and thus generalizability) than both Amazon models. Therefore, **overfitting is not the only factor that causes a model to leak information about its training dataset**. The structure and type of the model also contribute to the problem.

In Figure 11, we look deeper into the factors that contribute to attack accuracy per class. We analyze generalizability of the models using the (train-test) accuracy gap metric and the fraction of the training data that belongs to each class. The (train-test) accuracy gap is the difference between the accuracy of the target model on its training and test data. Similar metrics are used in the literature for measuring how overfitted a model is [14]. We compute this metric for each class. Bigger gaps indicate that the model is overfitted on its training data for the corresponding class. As the plots show and as expected, the bigger (train-test) accuracy gap is associated with higher precision of membership inference.

G. Effect of the number of classes and the amount of training data per class

The number of output classes of the target model contributes to how much the model leaks because the more classes, the more signals about the internal state of the model are available to the attacker. This is one of the reasons why the results in Fig. 5 are better for CIFAR-100 than for CIFAR-10. The CIFAR-100 model is also more overfitted to its training dataset. For the same number of training records per class, the attack performs better against CIFAR-100 than against CIFAR-10. For example, compare CIFAR-10 when the size of the training dataset is $2,000$ with CIFAR-100 when the size of the training dataset is $20,000$. The average number of data records per class is 200 in both cases, but the attack accuracy is much better (close to 1) for CIFAR-100.

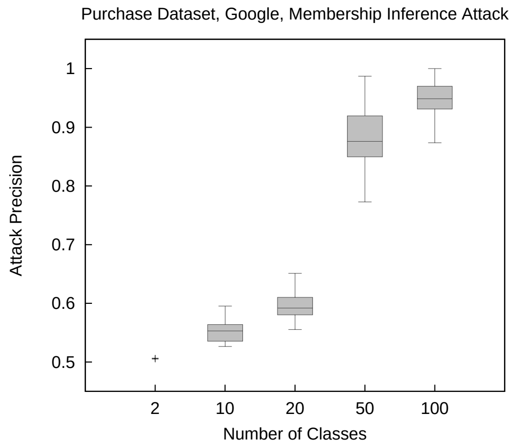

Fig. 10. Precision of the membership inference attack on different purchase classification models trained on the Google platform. The boxplots show the distribution of precision over different classification tasks (with different number of classes).

To quantify the effect that the number of classes has on the accuracy of the attack, we trained target models on the Google platform using the purchase dataset with $\{2, 10, 20, 50, 100\}$ classes. Figure 10 shows the distribution of attack precision for each model. Models with fewer classes leak less information about their members. As the number of classes increases, the model needs to extract more distinctive features from the data to be able to classify inputs with high accuracy. This means that models with more output classes need to remember more about their training data, hence higher leakage.

Figure 11 shows the relationship between the amount of training data per class and the accuracy of membership inference. This relationship is more complex, but, in general, the more data in the training dataset associated with a given class, the lower the attack precision for that class.

Table II shows the attack accuracy on Google-trained models. For the MNIST dataset, the training accuracy of the resulting target model is 0.984 and its test accuracy is 0.928. The overall precision of the membership inference attack is 0.517, which is just slightly above random guessing. The lack of randomness in the training data for each class and the small number of classes contributes to the low accuracy of the attack.

For the Adult dataset, the training accuracy of the resulting target model is 0.848 and its test accuracy is 0.842. The overall precision is 0.503, which is equivalent to random guessing. There could be two reasons for why membership inference fails against this model. First, the model is not overfitted (its test and train accuracies are almost the same). Second, the model is a binary classifier, which means that the attacker has to distinguish members from non-members by observing the behavior of the model on essentially 1 signal, since the two outputs are complements. This is not enough for our attack to extract useful membership information from the model.

Dataset	*Training Accuracy*	*Testing Accuracy*	*Attack Precision*
Adult	0.848	0.842	0.503
MNIST	0.984	0.928	0.517
Location	1.000	0.673	0.678
Purchase (2)	0.999	0.984	0.505
Purchase (10)	0.999	0.866	0.550
Purchase (20)	1.000	0.781	0.590
Purchase (50)	1.000	0.693	0.860
Purchase (100)	0.999	0.659	0.935

TABLE II
ACCURACY OF THE GOOGLE-TRAINED MODELS AND THE CORRESPONDING ATTACK PRECISION.

VIII. WHY OUR ATTACKS WORK

Table II shows the relationship between attack accuracy and test/train accuracy of the target models. Figure 12 also illustrates how the target models' outputs distinguish members from non-members (this is the information that our attack exploits). The plots show, for different datasets, how the model's outputs are different for the inputs that were in its training dataset versus the inputs that were not. Specifically, we look at the accuracy of the model in predicting the right label, as well as the prediction uncertainty of the model. The model accuracy for class i is the probability that it classifies an input with label i as i. Prediction uncertainty is the normalized entropy of the model's prediction vector ($\frac{-1}{\log(n)} \sum_i p_i \log(p_i)$, where p_i is the probability that the input belongs to class i, and n is the number of classes). The plots show that there is a clear difference between the output (both accuracy and uncertainty) of the model on the member inputs versus the non-member inputs, for the cases where our inference attack is successful.

Success of membership inference is directly related to (1) generalizability of the target model, and (2) diversity of its training data. If the model overfits and does not generalize well to data records beyond its training dataset, or if the training data is not representative, the model leaks information about its training data. We quantify this relationship in Fig. 11. From the machine learning perspective, overfitting is a well-known problem. It is harmful because it results in models that lack high predictive power. In this paper, we show another harm of overfitting: the leakage of sensitive information about the training data.

As we explained in Section VII, overfitting is not the only reason why our inference attacks work. Different machine learning models, due to their different structures, "remember" different amounts of information about their training datasets. This leads to different amounts of information leakage even if the models are overfitted to the same degree (e.g., see Table I).

IX. MITIGATION

As explained in Section VIII, multiple factors contribute to machine learning models leaking information about their training datasets. Overfitting is an important (but not the only) reason. Of course, overfitting is a canonical problem in machine learning research because it limits the predictive

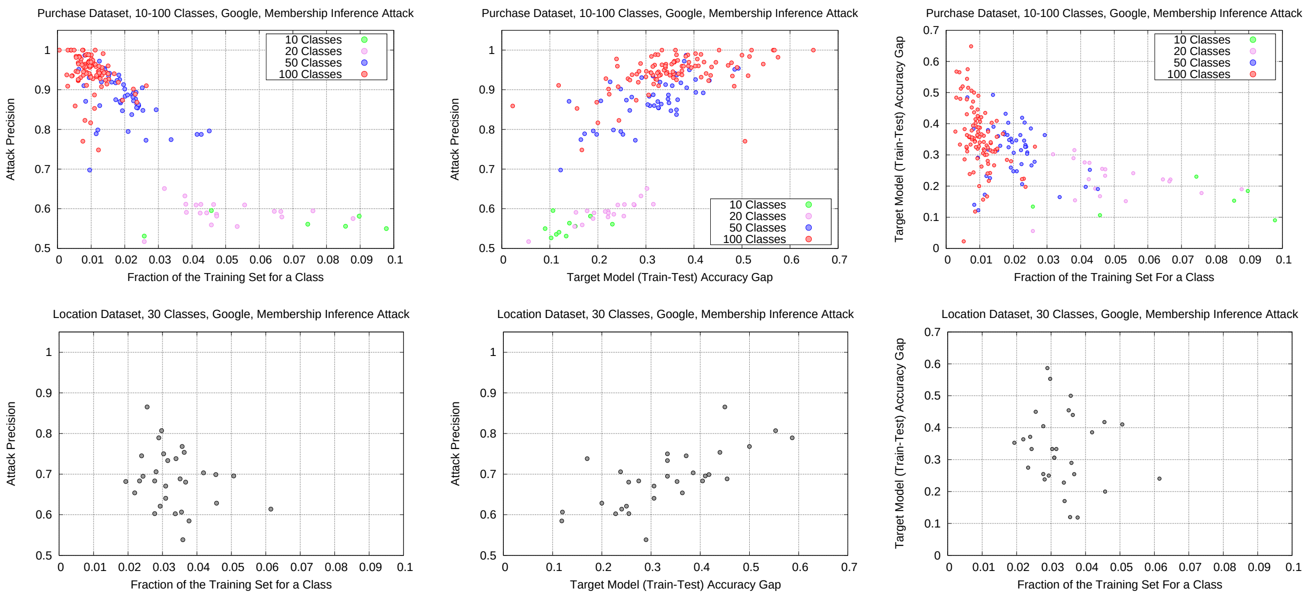

Fig. 11. Relationship between the precision of the membership inference attack on a class and the (train-test) accuracy gap of the target model, as well as the fraction of the training dataset that belongs to this class. Each point represent the values for one class. The (train-test) accuracy gap is a metric for generalization error [14] and an indicator of how overfitted the target model is.

power and generalizability of models. This means that instead of the usual tradeoff between utility and privacy, machine learning research and privacy research have similar objectives in this case. Regularization techniques, such as dropout [26], can help defeat overfitting. It has been shown that dropout can also strengthen privacy guarantees in a simple neural network [18]. Regularization techniques are also used for objective perturbation in empirical risk minimization in order to develop differentially private machine learning models [7].

(Ideal) well-regularized models should not leak much information about their training data, and our attack can serve as a metric to quantify this. Also, models with a trivial structure (e.g., XOR of some input features) generalize to the entire universe and do not leak information.

Our attacks are less effective if the attacker's access to the full output of the target model is strictly limited—for example, instead of the confidence scores for all classes, the model reports only the most likely class. Availability of the entire prediction vector highly depends on the needs of the application that uses the model, but in many cases applications need full scores (e.g., to classify an input that belongs to several classes). In practice, existing machine-learning-as-service platforms return the full vector of precise confidence scores.

If the training process is differentially private [10], the probability of producing a particular model from a training dataset that includes a given record is ϵ-close to the probability of producing the same model when this record is not included in the training dataset. Differentially private models are, by definition, secure against membership inference attacks of the kind developed in this paper, since our attacks operate solely on the outputs of the model, without any auxiliary information. One obstacle to designing differentially private models is that they may significantly reduce the model's prediction accuracy for small ϵ values. In Section II, we survey some of the related work in this area.

In the case of machine learning as a service, platform operators such as Google and Amazon have significant responsibility to the users of their services. In their current form, these services simply accept the data, produce a model of unknown type and structure, and return an opaque API to this model that data owners use as they see fit, without any understanding that by doing so, they may be leaking out their data. Learning services do not inform the users about the risks of overfitting or the harm that may result from models trained on inadequate datasets (for example, with unrepresentative records or too few representatives for certain classes).

Instead, when adaptively choosing a model for a user-supplied dataset, services such as Google Prediction API and Amazon ML should take into account not only the accuracy of the model but also the risk that it will leak information about its training data. Furthermore, they need to explicitly warn users about this risk and provide more visibility into the model and the methods that can be used to reduce the leakage. Our inference attacks can be used as metrics to quantify leakage from a specific model, and also to measure the effectiveness of future privacy protection techniques deployed by machine-learning services.

X. Conclusions

We have designed, implemented, and evaluated the first membership inference attack against machine learning models, notably black-box models trained in the cloud using Google Prediction API and Amazon ML. Our attack is the first general,

Fig. 12. Classification uncertainty and accuracy of the target model for the members of its training dataset vs. non-members, visualized for several sample classes. The difference between the member and non-member output distributions is among the factors that our attack exploits to infer membership. The accuracy of our attack is higher for the models where the two distributions are more distinguishable (See Table II).

quantitative approach to understanding how machine learning models leak information about their training datasets. When selecting the type of the model to train or a cloud-based learning service to use, our attack can be used as one of the metrics for choosing between models.

Our key technical innovation is the shadow training technique that trains an attack model to distinguish the target model's outputs on members vs. non-members of its training dataset. We demonstrate that shadow models used in this attack can be effectively created using synthetic or noisy data. In the case of synthetic data generated from the target model itself, the attack does not require any prior knowledge about the distribution of the target model's training data.

In all of our experiments, the classification models we attack are the types of models that many apps use to profile and classify users. Therefore, our results have substantial privacy implications. We believe that similar attacks can be performed against models trained on even more sensitive (e.g., clinical and biomedical) data, where our techniques can be used to evaluate the privacy risks of machine learning.

Acknowledgments

Big thanks to Congzheng Song for running model inversion on convolutional neural networks trained on the CIFAR-10 dataset and producing Figure 1.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.

[2] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[3] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Privacy-preserving ECG classification with branching programs and neural networks," *Trans. Info. Forensics and Security*, vol. 6, no. 2, pp. 452–468, 2011.

[4] J. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," *J. Biomed. Informatics*, vol. 50, pp. 234–243, 2014.

[5] J. Calandrino, A. Kilzer, A. Narayanan, E. Felten, and V. Shmatikov, "You might also like: Privacy risks on collaborative filtering," in *S&P*, 2011.

[6] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *NIPS*, 2009.

[7] K. Chaudhuri, C. Monteleoni, and A. Sarwate, "Differentially private empirical risk minimization," *JMLR*, vol. 12, pp. 1069–1109, 2011.

[8] I. Diakonikolas, M. Hardt, and L. Schmidt, "Differentially private learning of structured discrete distributions," in *NIPS*, 2015.

[9] W. Du, Y. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification." in *SDM*, 2004.

[10] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 338–340.

[11] C. Dwork and M. Naor, "On the difficulties of disclosure prevention in statistical databases or the case for differential privacy," *J. Privacy and Confidentiality*, vol. 2, no. 1, pp. 93–107, 2010.

[12] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *CCS*, 2015.

[13] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: an end-to-end case study of personalized warfarin dosing," in *USENIX Security*, 2014.

[14] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *arXiv:1509.01240*, 2015.

[15] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[16] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genetics*, vol. 4, no. 8, 2008.

[17] G. Jagannathan and R. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *KDD*, 2005.

[18] P. Jain, V. Kulkarni, A. Thakurta, and O. Williams, "To drop or not to drop: Robustness, consistency and differential privacy properties of dropout," *arXiv:1503.02031*, 2015.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[20] M. J. Kusner, J. R. Gardner, R. Garnett, and K. Q. Weinberger, "Differentially private bayesian optimization," in *ICML*, 2015.

[21] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *CRYPTO*, 2000.

[22] F. McSherry, "Statistical inference considered harmful," *https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14.md*, 2016.

[23] B. Rubinstein, P. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for SVM learning," *J. Privacy and Confidentiality*, vol. 4, no. 1, p. 4, 2012.

[24] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool," *Nature genetics*, vol. 41, no. 9, pp. 965–967, 2009.

[25] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security*, 2016.

[28] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD*, 2002.

[29] J. Vaidya, M. Kantarcıoğlu, and C. Clifton, "Privacy-preserving naive bayes classification," *VLDB*, vol. 17, no. 4, pp. 879–898, 2008.

[30] M. Wainwright, M. Jordan, and J. Duchi, "Privacy aware learning," in *NIPS*, 2012.

[31] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," *arXiv:1412.6181*, 2014.

[32] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM TIST*, vol. 7, no. 3, p. 30, 2016.

[33] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *VLDB*, vol. 5, no. 11, pp. 1364–1375, 2012.

[34] J. Zhu and T. Hastie, "Kernel logistic regression and the import vector machine," in *NIPS*, 2001.