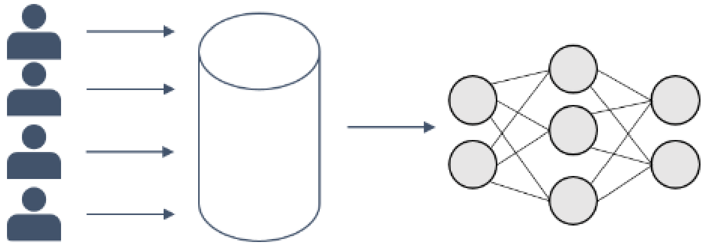
Model Inversion Attacks

Ref:

1. https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-fredrikson-privacy.pdf
2. https://franziska-boenisch.de/posts/2020/12/model-inversion/

Attack Against Neural Network

- Imagine, you would like to train a classifier.
- Usually, you start with some (potentially sensible) training data, i.e. some data features X and corresponding class labels y.
- pick an algorithm, e.g. a neural network (NN), and then you use your training data to make the model learn to map from X to y
- This mapping should generalize well, such that your model is also able to predict the correct labels for, so far, unseen data X.


A typical ML workflow.

Model inversion (MI) attacks are aimed at reconstructing training data from model parameters.

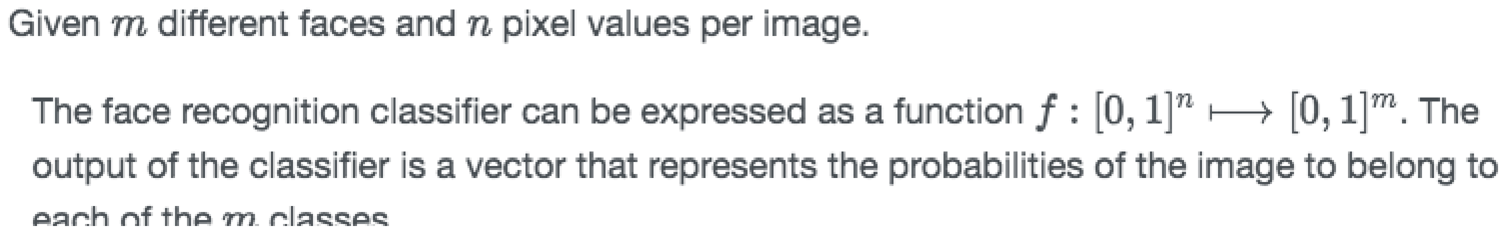
Model Inversion Attacks

proposed by Fredrikson et al. in 2015.
[https://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf][3]

face classifier trained on black and white images of 40 different individuals’ faces.

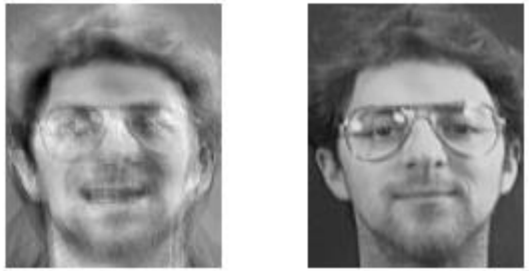
The data features X : individual image pixels that can take continuous values in the range of [0,1]

With this large number of different pixel values combinations over an image, it is inefficient to brute-force a reconstruction over all possible images in order to identify the most likely one(s).



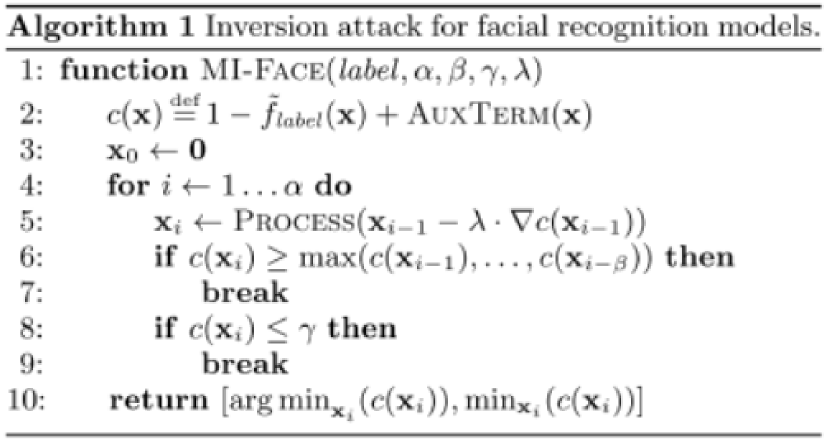
Given m different faces and n pixel values per image.

The face recognition classifier can be expressed as a function $f : [0,1]^n \longmapsto [0,1]^m$. The output of the classifier is a vector that represents the probabilities of the image to belong to each of the m classes



From [3]

The authors define a cost function $c(x)$ concerning f in order to do the model inversion. Starting with a candidate solution image x_0, the gradients of the cost function are calculated. Then, *gradient descent* is applied, and x_0 is transformed iteratively in each epoch i according to the gradients in order to minimize the cost function. After a minimum is found, the transformed x_i can be returned as the solution of the model inversion.



Algorithm 1 Inversion attack for facial recognition models.

```
function MI-FACE($label, \alpha, \beta, \gamma, \lambda$)
    $c(\mathbf{x}) \overset{\text{def}}{=} 1 - \tilde{f}_{label}(\mathbf{x}) + \text{AUXTERM}(\mathbf{x})$
    $\mathbf{x}_0 \leftarrow \mathbf{0}$
    for $i \leftarrow 1 \ldots \alpha$ do
        $\mathbf{x}_i \leftarrow \text{PROCESS}(\mathbf{x}_{i-1} - \lambda \cdot \nabla c(\mathbf{x}_{i-1}))$
        if $c(\mathbf{x}_i) \geq \max(c(\mathbf{x}_{i-1}), \ldots, c(\mathbf{x}_{i-\beta}))$ then
            break
        if $c(\mathbf{x}_i) \leq \gamma$ then
            break
    return $[\arg\min_{\mathbf{x}_i}(c(\mathbf{x}_i)), \min_{\mathbf{x}_i}(c(\mathbf{x}_i))]$
```

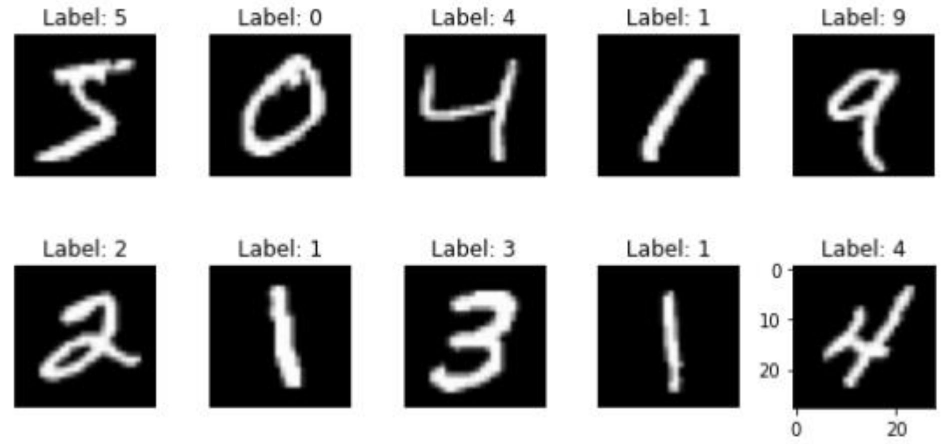
Let's have a concrete look at the algorithm specified in the Fredrikson paper with the parameters being: $label$: label of the class we want to inverse, α: number of iterations to calculate, β: a form of patience: if the cost does not decrease within this number of iterations, the algorithm stops, γ: minimum cost threshold, λ: learning rate for the gradient descent, and $AUXTERM$: a function that uses any available auxiliary information to inform the cost function. (In the case of simple inversion, there exists no auxiliary information, i.e. $AUXTERM = 0$ for all x.)

Example Code

https://github.com/Trusted-AI/adversarial-robustness-toolbox

Dataset:

MNIST dataset which consists of 60.000 training and 10.000 test black and white images of the digits from 0 to 9 with the corresponding labe



The first ten digits in the MNIST training dataset with corresponding labels.

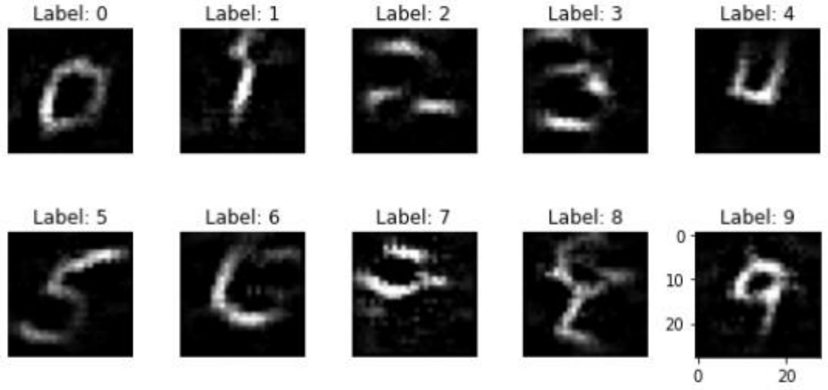
disable the eager execution - For ART framework

`Use KerasClassifier` as a wrapper.

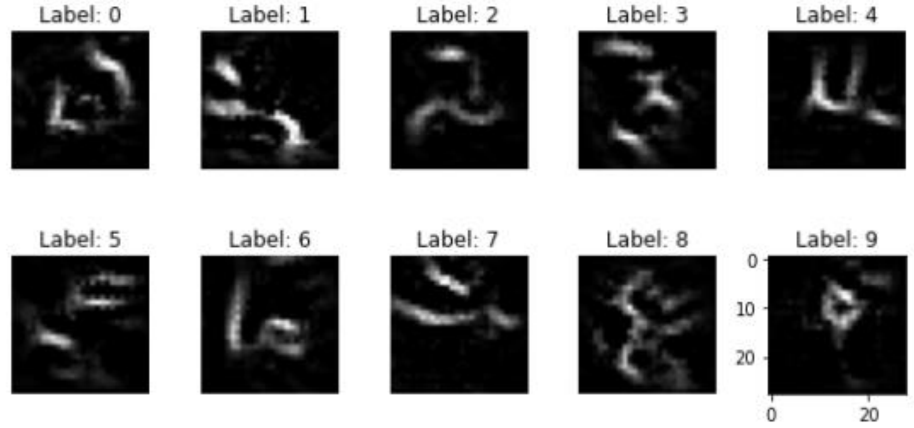
- Take a model
- Compile it with your chosen parameters
- Start the Attack

IBM-ART specifies array (x and y).

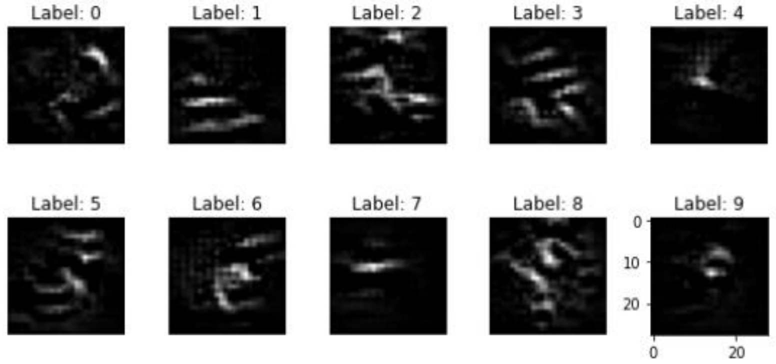
X: contains the initial input to the classifier under attack for each class label.
[not specified set as 0]



1 epoch of training: Restored training data class representations through model inversion.



10 epochs of training: Restored training data class representations through model inversion.



100 epochs of training: Restored training data class representations through model inversion.

Attack against linear regression model f

Predict:

a real-valued suggested initial dose of the drug Warfarin using a feature vector consisting of patient demographic information, medical history, and genetic markers.

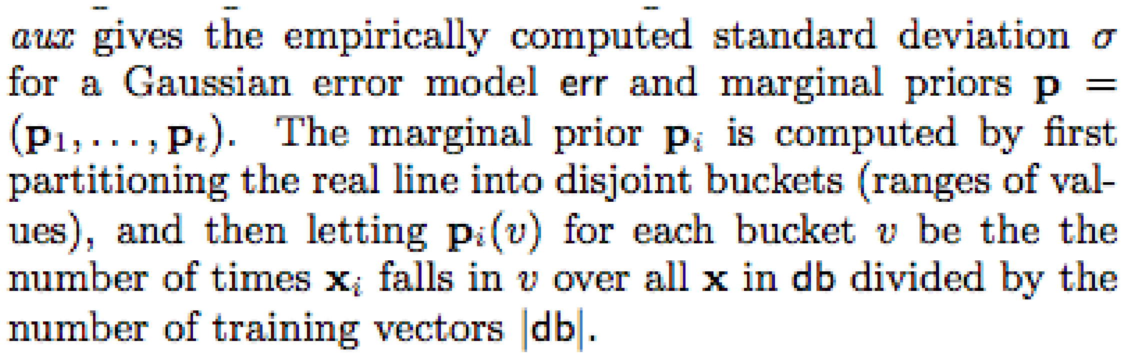
The sensitive attribute was considered to be the genetic marker (first feature x1.)

given white-box access to f

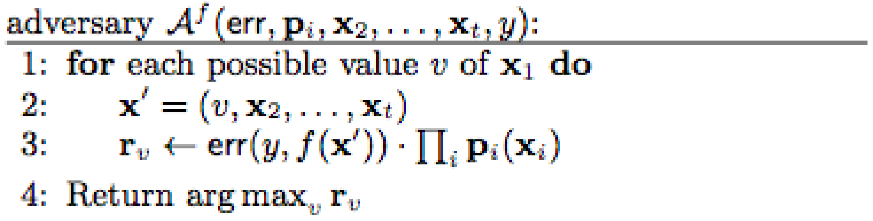
$\mathsf{side}(\mathbf{x}, y) \overset{\text{def}}{=} (\mathbf{x}_2, \ldots, \mathbf{x}_t, y)$ for a patient instance $(\mathbf{x}, y)$,

infer the patient's genetic marker $\mathbf{x}_1$.

Auxiliary Information



aux gives the empirically computed standard deviation σ for a Gaussian error model err and marginal priors $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_t)$. The marginal prior $\mathbf{p}_i$ is computed by first partitioning the real line into disjoint buckets (ranges of values), and then letting $\mathbf{p}_i(v)$ for each bucket v be the the number of times $\mathbf{x}_i$ falls in v over all $\mathbf{x}$ in db divided by the number of training vectors $|\mathsf{db}|$.



```
adversary $\mathcal{A}^f(\text{err}, \mathbf{p}_i, \mathbf{x}_2, \ldots, \mathbf{x}_t, y)$:
1: for each possible value $v$ of $\mathbf{x}_1$ do
2:     $\mathbf{x}' = (v, \mathbf{x}_2, \ldots, \mathbf{x}_t)$
3:     $\mathbf{r}_v \leftarrow \text{err}(y, f(\mathbf{x}')) \cdot \prod_i \mathbf{p}_i(\mathbf{x}_i)$
4: Return $\arg\max_v \mathbf{r}_v$
```

Gaussian error model will penalize x1 that force the prediction to be far from the given label y.

- **Least biased maximum a posteriori (MAP) estimate for x1 given the available information.**

- **Thus, it minimizes the adversary’s misprediction rate.**

Limitation

Not suitable when feature space is huge

Inversion attacks on decision trees

Why Decision Tree

model is used on a wide range of data

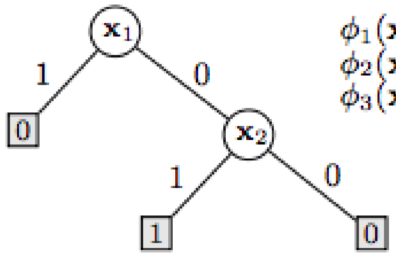
often favored because tree-structured rules are easy for users to understand.

[There are two types of decision trees common in the literature: classification (where the class variable is discrete), and regression (where the class variable is continuous).]

We focus on classification trees

Region

A decision tree model recursively partitions the feature space into disjoint regions R1, . . . , Rm. Predictions are made for an instance (x, y) by finding the region containing x, and returning the most likely value for y observed in the training data within that region.



$$\phi_1(\mathbf{x}) = \mathbf{x}_1 \qquad w_1 = 0$$
$$\phi_2(\mathbf{x}) = (1-\mathbf{x}_1)(\mathbf{x}_2) \qquad w_2 = 1$$
$$\phi_3(\mathbf{x}) = (1-\mathbf{x}_1)(1-\mathbf{x}_2) \qquad w_3 = 0$$

characterize trees mathematically as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}), \text{ where } \phi_i(\mathbf{x}) \in \{0,1\}$$

where each *basis function* ϕ_i is an indicator for R_i,

w_i corresponds to the most common response observed in the training set within R_i.

Classification:

$$\arg\max_j \left(\sum_{i=1}^{m} w_i[j]\phi_i(\mathbf{x})\right)$$

Confidences :

$$\tilde{f}(\mathbf{x}) \quad = \quad \left[\frac{w_{i^*}[1]}{\sum_i w_1[i]}, \ldots, \frac{w_{i^*}[|Y|]}{\sum_i w_m[i]}\right]$$

Decision tree APIs

Few allows users to publish trees in either black-box or white-box mode

[Eg: BigML]

In black-box mode, other users are only allowed to query the decision tree for predictions using a API.

In **both settings:**

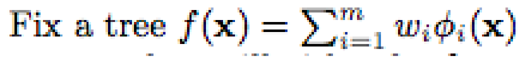
- **the adversary has access to marginal priors for each feature** of the training set ,
- Access to a **confusion matrix C** for which $C_{i,j}$ gives the number of training instances with $y = i$ for which the model predicted label j.

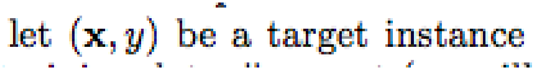
White-box mode(additionally):

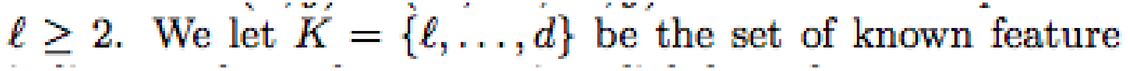
- users are allowed to see the **internal structure of the tree**, as well as download.

- In the white-box setting, the attacker also has **access to a count n_i of the number of training set instances that match path φ_i** in the tree.

Objective of Attack

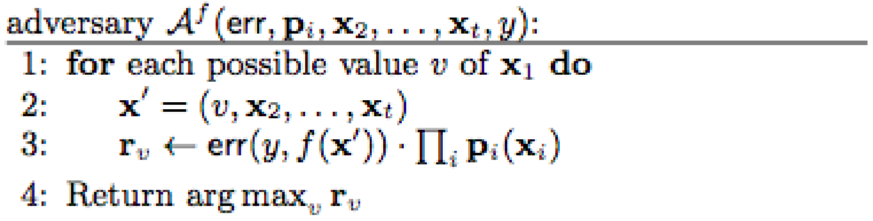

Fix a tree $f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$


let $(\mathbf{x}, y)$ be a target instance


$\ell \geq 2$. We let $K = \{\ell, \dots, d\}$ be the set of known feature

The adversary should output a value that maximizes for x1 [unknown sensitive feature]]

Black-box MI: Revisit the same algorithm

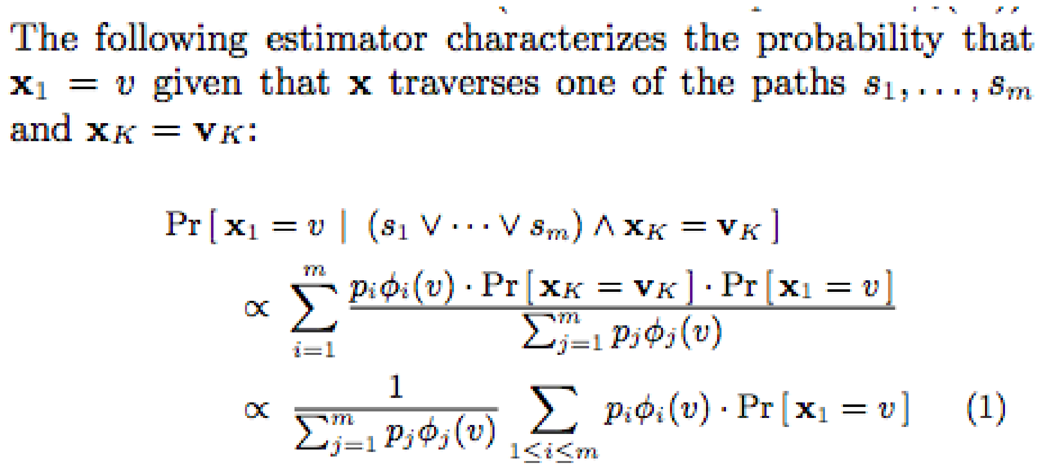


```
adversary $\mathcal{A}^f(\mathsf{err}, \mathbf{p}_i, \mathbf{x}_2, \ldots, \mathbf{x}_t, y)$:
1: for each possible value $v$ of $\mathbf{x}_1$ do
2:     $\mathbf{x}' = (v, \mathbf{x}_2, \ldots, \mathbf{x}_t)$
3:     $\mathbf{r}_v \leftarrow \mathsf{err}(y, f(\mathbf{x}')) \cdot \prod_i \mathbf{p}_i(\mathbf{x}_i)$
4: Return $\arg\max_v \mathbf{r}_v$
```

- decision tree models we consider produce discrete outputs
- the error model information: confusion matrix as opposed to the standard deviation of a Gaussian.

$$\mathsf{err}(y, y') \propto \Pr[\, f(\mathbf{x}) = y' \mid y \text{ is the true label} \,]$$

White-box MI



The following estimator characterizes the probability that $\mathbf{x}_1 = v$ given that $\mathbf{x}$ traverses one of the paths $s_1, \ldots, s_m$ and $\mathbf{x}_K = \mathbf{v}_K$:

$$\begin{aligned}
&\Pr\left[\mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K\right] \\
&\quad \propto \sum_{i=1}^{m} \frac{p_i \phi_i(v) \cdot \Pr\left[\mathbf{x}_K = \mathbf{v}_K\right] \cdot \Pr\left[\mathbf{x}_1 = v\right]}{\sum_{j=1}^{m} p_j \phi_j(v)} \\
&\quad \propto \frac{1}{\sum_{j=1}^{m} p_j \phi_j(v)} \sum_{1 \le i \le m} p_i \phi_i(v) \cdot \Pr\left[\mathbf{x}_1 = v\right] \qquad (1)
\end{aligned}$$

The adversary then outputs a value for v that maximizes (1) as a guess for x1.

White-box MI

$$\begin{aligned}
&\Pr\left[\,\mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K\,\right] \\
&\quad\propto \sum_{i=1}^{m} \frac{p_i\phi_i(v)\cdot\Pr\left[\,\mathbf{x}_K = \mathbf{v}_K\,\right]\cdot\Pr\left[\,\mathbf{x}_1 = v\,\right]}{\sum_{j=1}^{m} p_j\phi_j(v)} \\
&\quad\propto \frac{1}{\sum_{j=1}^{m} p_j\phi_j(v)} \sum_{1\le i\le m} p_i\phi_i(v)\cdot\Pr\left[\,\mathbf{x}_1 = v\,\right] \qquad (1)
\end{aligned}$$

Line 1: pi denote ni/N.

The known values $\mathbf{x}_K$ induce a set of paths $S = \{s_i\}_{1\le i\le m}$: $S = \{(\phi_i, n_i) \mid \exists \mathbf{x}' \in \mathbb{R}^d \,.\, \mathbf{x}'_K = \mathbf{x}_K \wedge \phi_i(\mathbf{x}')\}$. Each path corresponds to a basis function ϕ_i and a sample count n_i.

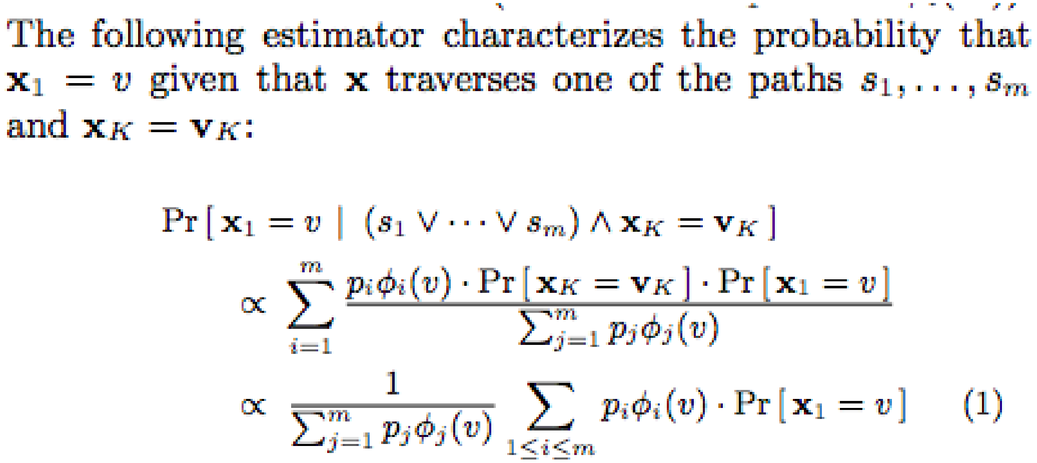
White-box MI

$\phi_i(v)$

indicator function $\mathbb{I}(\exists \mathbf{x}' \in \mathbb{R}^d \;.\; \mathbf{x}'_1 = v \wedge \phi_i(\mathbf{x}'))$.

$$\begin{aligned}
&\Pr\left[\, \mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K \,\right] \\
&\quad \propto \sum_{i=1}^{m} \frac{p_i \phi_i(v) \cdot \Pr\left[\, \mathbf{x}_K = \mathbf{v}_K \,\right] \cdot \Pr\left[\, \mathbf{x}_1 = v \,\right]}{\sum_{j=1}^{m} p_j \phi_j(v)} \\
&\quad \propto \frac{1}{\sum_{j=1}^{m} p_j \phi_j(v)} \sum_{1 \le i \le m} p_i \phi_i(v) \cdot \Pr\left[\, \mathbf{x}_1 = v \,\right] \qquad (1)
\end{aligned}$$

White-box MI



The following estimator characterizes the probability that $\mathbf{x}_1 = v$ given that $\mathbf{x}$ traverses one of the paths $s_1, \ldots, s_m$ and $\mathbf{x}_K = \mathbf{v}_K$:

$$\begin{aligned}
&\Pr\left[\mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K\right] \\
&\quad \propto \sum_{i=1}^{m} \frac{p_i \phi_i(v) \cdot \Pr\left[\mathbf{x}_K = \mathbf{v}_K\right] \cdot \Pr\left[\mathbf{x}_1 = v\right]}{\sum_{j=1}^{m} p_j \phi_j(v)} \\
&\quad \propto \frac{1}{\sum_{j=1}^{m} p_j \phi_j(v)} \sum_{1 \le i \le m} p_i \phi_i(v) \cdot \Pr\left[\mathbf{x}_1 = v\right] \qquad (1)
\end{aligned}$$

The adversary then outputs a value for v that maximizes (1) as a guess for x1.

Membership Inference Attacks

Ref

https://arxiv.org/pdf/1610.05820.pdf

given a data record and black-box access to a model, determine if the record was in the model's training dataset.

To answer the membership inference question

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

Shadow training

First, create multiple “shadow models” that imitate the behavior of the target model, but for which   the training datasets are known

Thus the ground truth about membership in these dataset are known.

then train the attack model on the labeled inputs and outputs of the shadow models.

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 potentially 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.]

Inference techniques are generic:

-not based on any particular dataset or model type

[Effective for Amazon ML and Google Prediction API.

without knowing the learning algorithms used, nor the architecture of the resulting models, since Amazon and Google don’t reveal this information to the customers.]

Confidence values

For any input data record, the model outputs the prediction vector of probabilities, one per class, that the record belongs to a certain class.

These probabilities are confidence values.

Attack Steps

Consider a set of labeled data records sampled from some population and partitioned into classes. W

assume that a machine learning algorithm is used to train a classification model that captures the relationship between the content of the data records and their labels.

assume that the attacker has query access to the model and can obtain the model’s prediction vector on any data record

Attacker’s Knowledge

attacker knows the format of the inputs and outputs of the model, including their number and the range of values they can take.

the attacker either

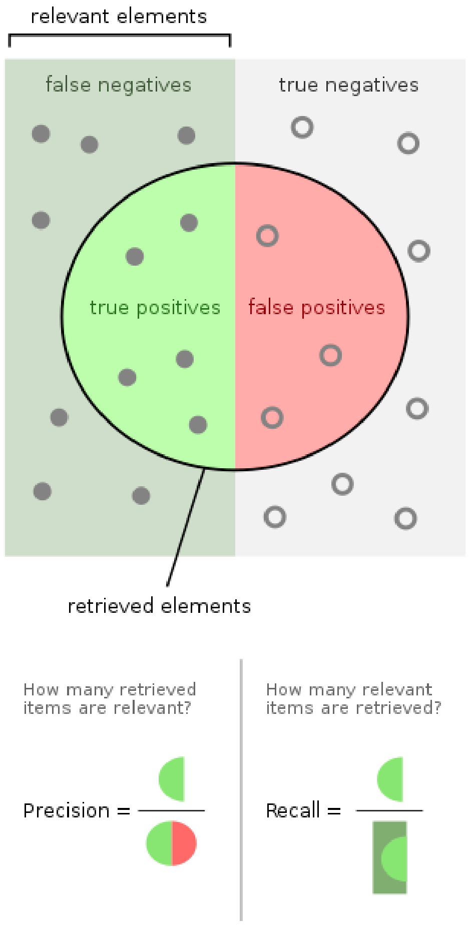
(1) knows the type and architecture of the machine learning model, as well as the training algorithm,

or (2) has black-box access to a machine learning oracle (e.g., a “machine learning as a service” platform) that was used to train the model. In the latter case, the attacker does not know a priori the model’s structure or meta-parameters

Attack Metric

Precision (what fraction of records inferred as members are indeed members of the training dataset) and

Recall (what fraction of the training dataset's members are correctly inferred as members by the attacker).



Overview of the attack

Observation:

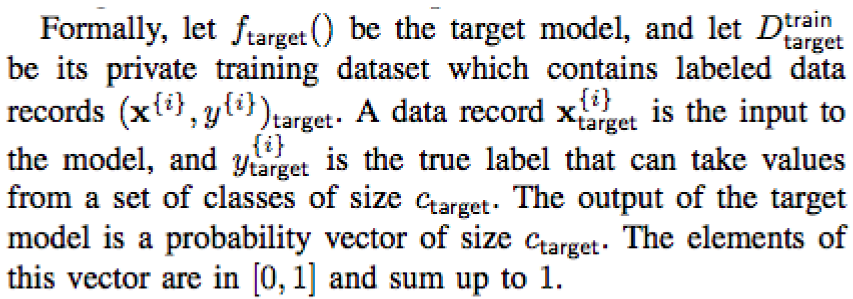
membership inference attack exploits the observation that machine learning models often behave differently on the data that they were trained on versus the data that they “see” for the first time.

Overfitting is a common reason but not the only one

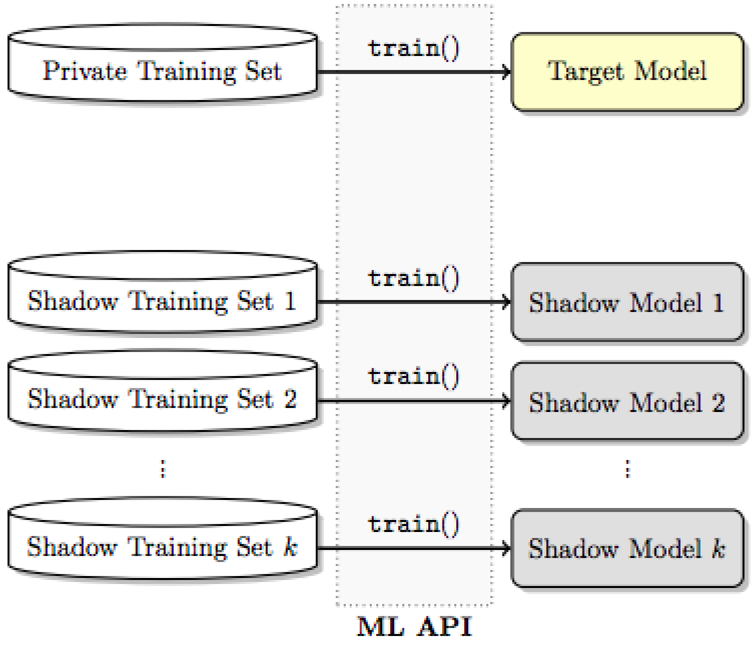
Objective of the attacker :

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.

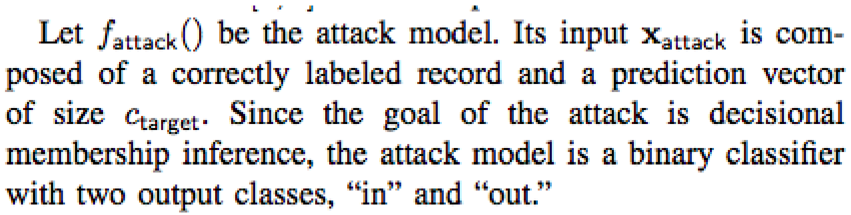
Target Model



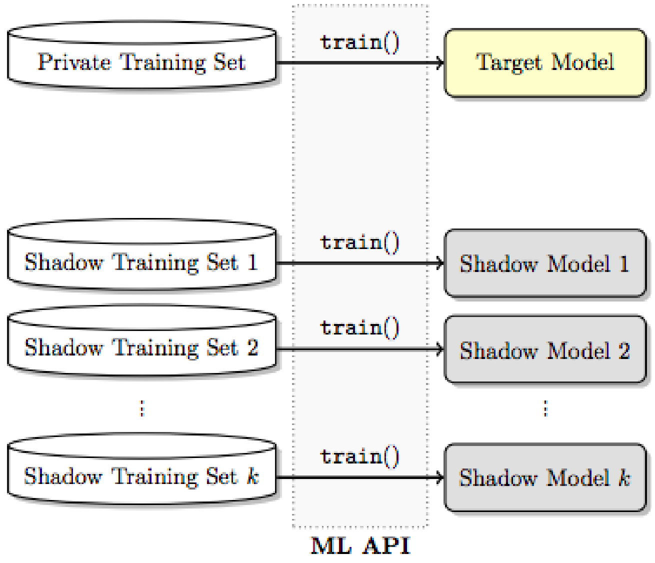
Formally, let $f_{\text{target}}()$ be the target model, and let $D^{\text{train}}_{\text{target}}$ be its private training dataset which contains labeled data records $(\mathbf{x}^{\{i\}}, y^{\{i\}})_{\text{target}}$. A data record $\mathbf{x}^{\{i\}}_{\text{target}}$ is the input to the model, and $y^{\{i\}}_{\text{target}}$ 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 elements of this vector are in $[0, 1]$ and sum up to 1.



Attack Model



Let $f_{\text{attack}}()$ be the attack model. Its input $\mathbf{x}_{\text{attack}}$ is composed of a correctly labeled record and a prediction vector of size c_{target}. Since the goal of the attack is decisional membership inference, the attack model is a binary classifier with two output classes, “in” and “out.”



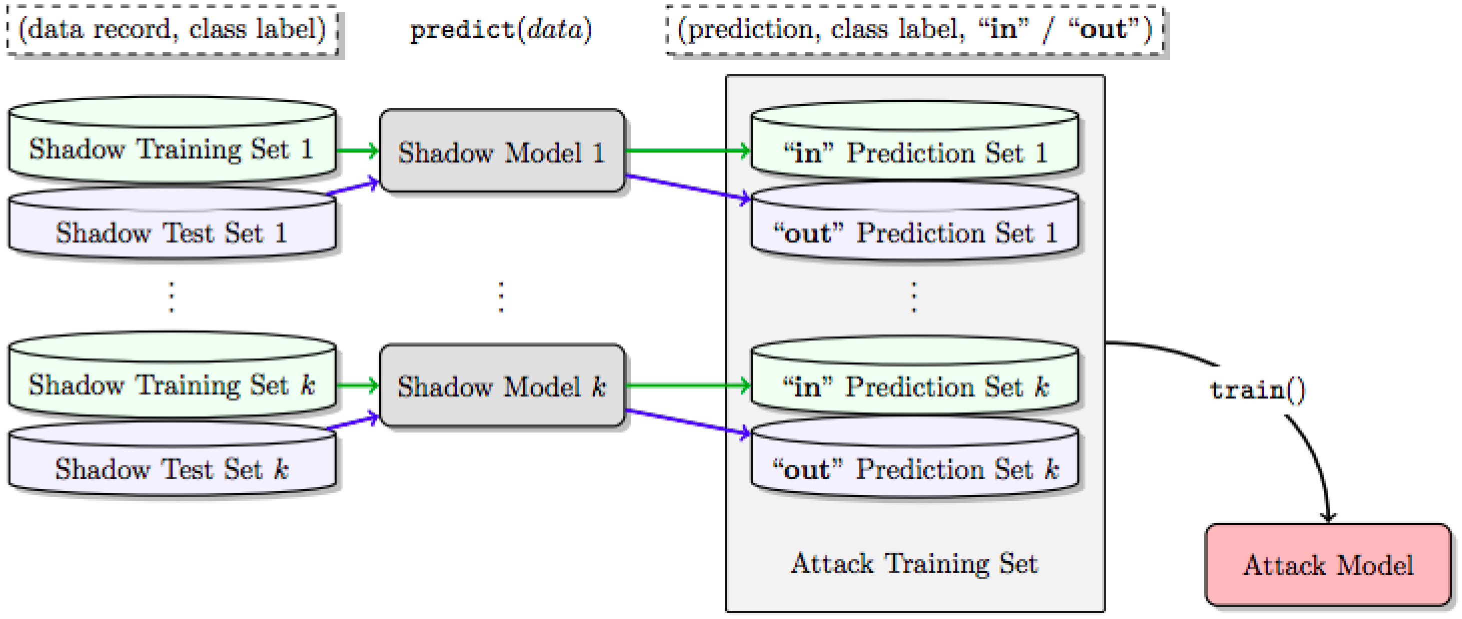
labeled record $(\mathbf{x}, y)$, we use the target model to compute the prediction vector $\mathbf{y} = f_{\text{target}}(\mathbf{x})$. The distribution of $\mathbf{y}$ (classification confidence values) depends heavily on the true class of $\mathbf{x}$. This is why we pass the true label y of $\mathbf{x}$ in addition to the model's prediction vector $\mathbf{y}$ to the attack model. Given how the probabilities in $\mathbf{y}$ are distributed around y, the attack model computes the membership probability $\Pr\{(\mathbf{x}, y) \in D^{\text{train}}_{\text{target}}\}$, i.e., the probability that $((\mathbf{x}, y), \mathbf{y})$ belongs to the "in" class or, equivalently, that $\mathbf{x}$ is in the training dataset of $f_{\text{target}}()$.

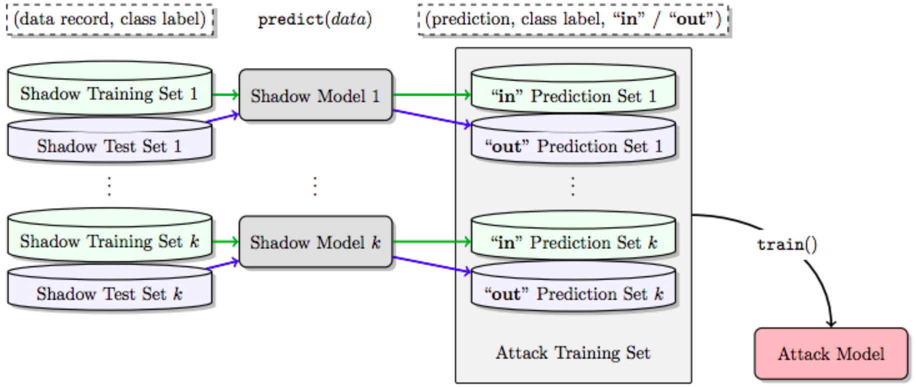
Shadow Models

The more shadow models, the more accurate the attack model will be.

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, more shadow models provide more training fodder for the attack model.


(data record, class label)
predict(data)
(prediction, class label, "in" / "out")
Shadow Training Set 1
Shadow Test Set 1
Shadow Model 1
"in" Prediction Set 1
"out" Prediction Set 1
Shadow Training Set k
Shadow Test Set k
Shadow Model k
"in" Prediction Set k
"out" Prediction Set k
Attack Training Set
train()
Attack Model



$(\mathbf{x}, y) \in D^{\text{train}}_{\text{shadow}^i}$, 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 records disjoint from the training set of the ith shadow model. Then, $\forall(\mathbf{x}, y) \in D^{\text{test}}_{\text{shadow}^i}$ 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, split $D^{\text{train}}_{\text{attack}}$ into c_{target} partitions, each associated with a different class label. For each label y, train a separate model that, given $\mathbf{y}$, predicts the in or out membership status for $\mathbf{x}$.