On Privacy in Machine Learning by Plausible Deniability

Guest Lecture at the Institute of Applied Statistics

Stefan Rass

Secure Systems Group, LIT Secure and Correct Systems Lab
Johannes Kepler University Linz
stefan.rass@jku.at

SS 2022
2022-03-17

On Privacy in Machine Learning by Plausible Deniability

Secure Systems Group, LIT Secure and Correct Systems Lab
Johannes Kepler University Linz
stefan.rass@jku.at

SS 2022
2022-03-17

Contents

Motivation: Co-Simulation for Security

- Former FFG project "ODYSSEUS": Security of interconnected critical infrastructures, including (but not limited to):
  - electricity
  - water
  - medical care and supplies
  - telecommunication
  - traffic
  - ...
- For many (not all) domains, we have simulation tools
- Each delivering accurate simulations of how environments respond to external stimuli by events/incidents (e.g., power shortages, road blockings upon accidents, ...)
- Incidents or events thereby have impacts over several infrastructures. Taking the infrastructures as connected via a graph topology (edges being interdependencies in a supply/demand relation), the incident percolates through the graph.
- We call this a "cascading effect"

Problems and a Solution Approach 1

The challenge:

- Simulation tools are, mostly, standalone software
- Difficult (if possible) to script, and interface with
- requires wrappers programmed around the simulator to connect with other simulators
- not substantially less effort than writing one's own simulation from scratch

The solution:

- Instead of interconnecting different simulators, resort to emulation...
- ...by training deep neural networks to mimic the response dynamics, learned from extensive simulation (data)

The problems:

- Where to get the data from? This, and simulation tools (if any), are in possession of critical infrastructure providers

Problems and a Solution Approach 2

- Classified, highly sensitive, information → strictly forbidden to give away
- Neither can a CI provider admit outside links to others (for security reason)

The solution proposal:

- Let the infrastructure provider run the simulation in its own premises, in high-security environments
- Do the training within these closed walls and give away only the trained deep-net

The CI provider's reply was concerned

The data we trained in is still sensitive and classified. How do we know that this information will not leak from a trained machine learning (ML) model?

⇒ this gave us a research question!

Research Question

- Obviously, given an ML model $f : \mathbb{R}^n \to \mathbb{R}$ trained upon a (huge) set of input-output pairs $(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, we can compute as many input-putput pairs of our own choice $\to$ training data is only confidential to some extent (e.g., measured by the "recall")
- Attacker could try to recover the training data from the ML model $f(\cdot, \mathbf{p}^*)$, and claim to have recovered a certain data set $T' = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, N\}$. This is an optimization problem:

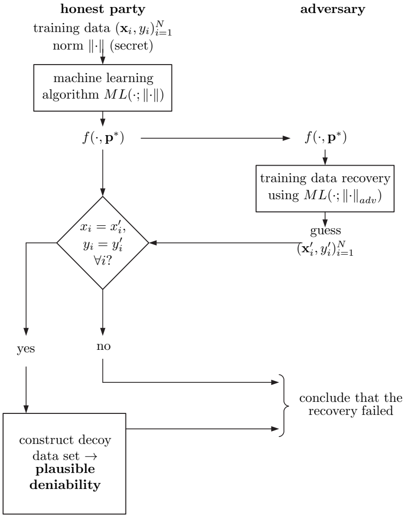
$$T' = \text{argmin} \left\| (y_i - f(\mathbf{x}_i, \mathbf{p}^*))_{i=1}^n \right\|$$

Research Question

Can we plausibly deny that the attacker's finding T' is correct (even if it was)?

As a workflow, plausible deniability can be interpreted to be a certain sequence of events, as shown on the right →

The goal of this work is showing that the honest party can succeed here!



Formalizing the Problem 1

- Let formally approach the training problem: given a family

$$ML = \left\{ f_{\mathbf{p}} : \mathbb{R}^m \to \mathbb{R} \;\middle|\; \mathbf{p} \in \mathbb{R}^d \right\},$$

  parameterized by some vector $\mathbf{p}$, the training algorithm is yet just another function $train : \mathbb{R}^{n\times(m+1)} \times \mathbb{R}^d \to ML$, mapping a training data matrix with n records $(\mathbf{x}_i, y_i) \in \mathbb{R}^{m+1}$.
- The training algorithm is an(other) optimization

$$\min \left\|(y_i - f(\mathbf{x}_i, \mathbf{p}))_{i=1}^n\right\| \text{ over } \mathbf{p}, \tag{1.1}$$

...because many of the usual error metrics are expressible as norms, such as:

1) Mean squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

2) Root mean squared error

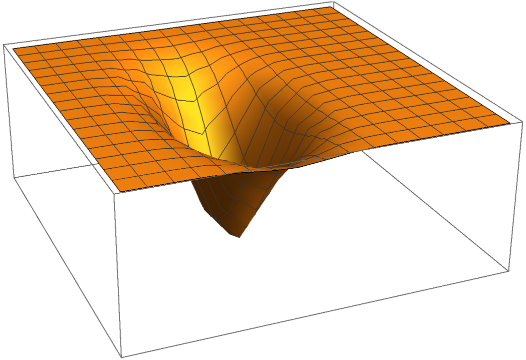
$$RMSE = \sqrt{MSE} = \frac{1}{\sqrt{n}}\|\mathbf{y} - \hat{\mathbf{y}}\|_2$$

3) Mean absolute error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| = \frac{1}{n}\cdot\|\mathbf{y} - \hat{\mathbf{y}}\|_1$$

Why Norms? 2

Also, norms are "more plausible" to argue, since trivial solutions would be obviously suspicious: the function below has a global minimum at some desired point. . . but very much looks (and in fact is) crafted towards this global optimum



- For deniability, it is already enough if the training function is not injective (in a strong sense):
- If every ML model $f \in ML$ has at least two pre-images T, T', i.e., training sets that would map to the same (target) model f, then whenever the adversary extracts T, we can claim the correct result to have been T' (and vice versa)
- A sufficient condition is Theorem 1.1.

Theorem 1.1 ([RKW+21])

Let the (unknown) training data come from a random source Z with entropy $H(Z)$ bits, and let the function f require (at least) k bits to encode, and assume that f has been trained from n unknown records.
If the number n exceeds

$$n > \frac{k}{H(Z)},$$

then any candidate training data extracted from f is deniable

Proof (Idea only): If a lot of L bits of training data map to a (smaller) ML model taking only $\ell < L$ bits to encode, there must be at least two different training sets mapping to the same ML model (pigeon hole principle) □

- What about smaller training sets that would, theoretically, fit into the size of the ML models description?
- We can extend the notion of plausible deniability to training sets of any size, if we manipulate the error metric in (1.1) accordingly.
- Let us:
  - Choose an arbitrary decoy data set $T' = \{(\mathbf{x}_1', y_1'), \ldots, (\mathbf{x}_n', y_n')\}$ to later claim having trained the given model $f(\cdot, \mathbf{p}^*)$ from it.
  - Compute the error vector $\mathbf{e} = (f(\mathbf{x}_i', \mathbf{p}^*) - y_i')_{i=1}^n$
  - And define a semi-norm $b(\mathbf{x}) := \|\mathbf{B} \cdot \mathbf{x}\|$, where the matrix $\mathbf{B}$ is chosen have exactly $\mathbf{x}$ as its nullspace ($\| \;\; \|$ is a (full) norm herein).
  - This ensures that $b(\mathbf{x}) = 0$ if $\mathbf{x} = \lambda \cdot \mathbf{e}$ for some $\lambda \in \mathbb{R}$, and $b(\mathbf{x}) > 0$ otherwise.
- This is close to what we want, but has a multitude of minima other than at the desired location $\mathbf{p}^*$, to which we have crafted the error vector $\mathbf{e}$.
- However, with some additional assumptions, we can assure local optimality at $\mathbf{p}^*$; this is Lemma 1.1.

Lemma 1.1 ([RKW+21])

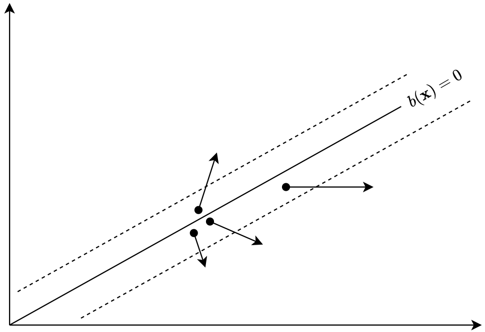
Let $f : \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}$ *be parameterized by a vector* $\mathbf{p} \in \mathbb{R}^d$ *and map an input value vector* $\mathbf{x}$ *to a vector* $\mathbf{y} = f(\mathbf{x}, \mathbf{p})$*. Let* $\mathbf{p}^* \in \mathbb{R}^d$ *be given as fixed, and let us pick arbitrary training data* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$*. Finally, define the error vector* $\mathbf{e} = (y_i - f(\mathbf{x}_i, \mathbf{p}^*))_{i=1}^n \in \mathbb{R}^n$*.*
Let for all $\mathbf{x}_i$ *the functions* $f(\mathbf{x}_i, \cdot)$ *be totally differentiable w.r.t.* $\mathbf{p}$ *at* $\mathbf{p} = \mathbf{p}^*$ *with derivative* $\mathbf{d}_i = D_{\mathbf{p}}(f(\mathbf{x}_i, \mathbf{p}))(\mathbf{p}^*) \in \mathbb{R}^d$*. Put all* $\mathbf{d}_i^\top$ *for* $i = 1, 2, \ldots, n$ *as rows into a matrix* $\mathbf{M} \in \mathbb{R}^{n \times d}$ *and assume that it satisfies the rank condition*

$$rank(\mathbf{M}|\mathbf{e}) > rank(\mathbf{M}). \tag{1.2}$$

Then, there exists a semi-norm $\|\cdot\|$ *on* $\mathbb{R}^n$ *such that* $\mathbf{p}^*$ *locally minimizes* $\|e(\mathbf{p}^*)\|$*, i.e., there is an open neighborhood* U *of* $\mathbf{p}^*$ *inside which* $\|e(\mathbf{p}^*)\| \leq \|e(\mathbf{p})\|$ *for all* $\mathbf{p} \in U$*.*

Proof (Sketch; idea only):

The rank condition essentially implies that any (small) displacement $\mathbf{p} \neq \mathbf{p}^*$ will lead outwards of span($\mathbf{e}$), and hence make the semi-norm $b(\mathbf{p}) > 0$.
This implies (local) optimality at the desired point $\mathbf{p}^*$, which is our target ML model.



□

Constructing a (full) Norm 1

- We can convert the semi-norm into a full toplogical norm, without additional requirements (only with slightly more effort on the proof):

Theorem 1.1 ([RKW+21])

Under the hypotheses of Lemma 1.1, there exists a norm $\|\cdot\|$ on $\mathbb{R}^n$ such that $\mathbf{p}^*$ locally minimizes $\|e(\mathbf{p})\|$ as a function of $\mathbf{p}$.

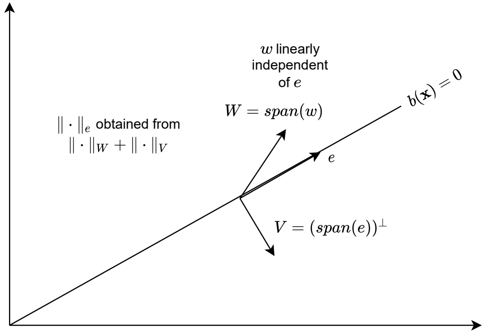
- Theorem 1.1 has several corollaries:
  - Generalization to vector-valued models f mapping into $\mathbb{R}^k \to$ Corollary 1.1; Appendix (S. 1-29)
  - Representation of the error metric in terms of a mean absolute error, rather than a "suspicious" norm $\to$ Corollary 1.2; Appendix (S. 1-30)

Proof (Sketch; idea only):

The sought norm will be

$$\|\mathbf{x}\| := \|\mathbf{x}\|_e + b(\mathbf{x}), \qquad (1.3)$$

with a norm $\|\cdot\|_e$ that depends on the vector $\mathbf{e}$. It is constructed from two other norms, one on the orthogonal subspace of $span(\mathbf{e})$, the other on a 1-dimensional complement space that is linearly independent of $span(\mathbf{e})$ $\rightarrow$



This norm $\|\cdot\|_e$ then (only) needs to satisfy $\|e(\mathbf{p}^*) - e(\mathbf{p})\|_e \leq b(e(\mathbf{p}))$ to preserve local optimality (this is the more difficult part of the proof). □

We demonstrate the construction as follows:

1) Instantiate a linear regression model with coefficients $\mathbf{p}^* = (\beta_0, \ldots, \beta_{d-1})$ chosen at random:
$$f(\mathbf{x}, \mathbf{p}^*) = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_{d-1} \cdot x_{d-1} + \varepsilon, \tag{1.4}$$
2) Sample from the linear model to get data that "fits well". This is the original training data, which is $\mathbf{x}_i \sim \mathcal{U}(\{1, 2, \ldots, 8\}^m)$, and $y_i := f(\mathbf{x}_i, \mathbf{p}^*) + \varepsilon$, for $i = 1, 2, \ldots, m$; with ε being random noise.
3) Then, generate *decoy* training data $\mathbf{X}'_i \sim \mathcal{U}(\{1, \ldots, 8\}^m)$, *and* another set of random, and hence unrelated, response values $Y'_i \sim \mathcal{U}(\{1, \ldots, 8\})$.
   Note that:
   - the decoy data is stochastically independent
   - the decoy's response variable y' has nothing to do with the decoy x'-values.

Experiment: Regression Model 2

4) Next, craft the norm as the proof of Theorem 1.1 prescribes – it is a constructive argument.
   This model is nice to use, since it admits a closed form expression for the Jacobian to check the hypothesis of Lemma 1.1.
5) And finally, let an optimizer run to re-create the model (1.4) using the crafted norm as error metric and the decoy data.

Results: Regression Model

original vector **p**	**p** as trained from decoy data T'
-0.57104	-0.56936
-1.53456	-1.53402
-2.45770	-2.45657
-2.12341	-2.12261
-1.26093	-1.25992
-1.91170	-1.91082

- This indicates that the idea and construction works quite well,
- so let us check another model of machine learning.

- Like before, we pick a random regression model, compute the log-odds, and pick random data as decoy to craft a norm to.
- The model is similar to the linear model, only has a sigmoid function $\sigma(x) = (1 + \exp(-x))^{-1}$ applied afterwards:

$$y = \sigma\left(\beta_0 + \boldsymbol{\beta}^\top \cdot \mathbf{x}\right), \tag{1.5}$$

  with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$ and $\mathbf{p} = (\beta_0, \boldsymbol{\beta})$.
- Like as for regression, the Jacobian can be worked out analytically (for Lemma 1.1.

Results: Logistic Regression 1

- The construction works, and the logistic regression model is recovered from the unrelated decoy data → Table 1
- However, in some cases, the optimizer drifts far off the desired location → Table 2
- Nonetheless, if the optimizer starts from the target $\mathbf{p}^*$, it does not move → Table 3
  This indicates (experimentally) that $\mathbf{p}^*$ is apparently a local optimum (as desired)

original vector $\mathbf{p}$	$\mathbf{p}$ as trained from decoy data T'	starting point
-36.452	-36.451	-36.408
16.448	16.447	16.504
13.043	13.044	13.045
24.545	24.546	24.571
40.418	40.419	40.489
-33.886	-33.887	-33.869

Table 1: Logistic Regression: Hitting the Target Model

original vector $\mathbf{p}$	$\mathbf{p}$ as trained from decoy data T'	starting point
-22.604	-7.6157e+03	-22.560
25.976	2.3044e+04	26.007
29.200	4.0411e+03	29.210
9.7599	1.8252e+04	9.7835
42.462	-7.5179e+04	42.481
-44.693	2.6841e+04	-44.674

Table 2: Logistic Regression: Missing the Target Model

Results: Logistic Regression 3

original vector **p**	**p** as trained from decoy data T'	starting point $= \mathbf{p}$
-44.774	-44.774	-44.774
-17.186	-17.186	-17.186
28.215	28.215	28.215
39.373	39.373	39.373
-39.419	-39.419	-39.419
-22.533	-22.533	-22.533

Table 3: Logistic Regression: No move if we start from **p**

Experiment: Neural Network

- With the same setup again, let us consider a feed-forward neural network

$$y = \sigma_\ell(\mathbf{W}_\ell \cdot \boldsymbol{\sigma}_{\ell-1}(\mathbf{W}_{\ell-1} \cdot \boldsymbol{\sigma}_{\ell-2}(\cdots \boldsymbol{\sigma}_1(\mathbf{W}_1 \cdot \mathbf{x}) \cdots))), \tag{1.6}$$

  where
  - ℓ is the total number of layers in the network,
  - each matrix $\mathbf{W}_i$ with $1 \leq i \leq \ell$ is the individual weighting between the output of the previous and input of the next layer, which row-wise gives the net value that goes into the activation functions, collected in the vector-valued function $\boldsymbol{\sigma}_i$ for the i-th layer.
  - we took $\sigma_i = (\tanh, \tanh, \ldots, \tanh)$ for all layers,
  - and let each inner layer have the same dimension, with the final function σ_ℓ outputting only a single real value.

Results: Neural Network

Long story short: Results were as for the logistic regression but further instructive

- Model could be successfully re-created from the decoy data
- But the solver, in more yet not all cases, drifted away from the target
- Taking a look at the eigenvalues of the (approximate) Hessian at $\mathbf{p}^*$, we found them to be in the range $< 3.1058 \times 10^{-3}$ up to $\approx 2.9067 \times 10^4$ (different for each experiment, since everything was initialized at random)
- This indicates that the basin of attraction seems to be a very “flat” ellipsis
- This also explains why the construction generally failed (in further experiments) where we applied a randomized optimization like stochastic gradient decent: the solver there very likely jumps out of the basin of atttraction and drifts elsewhere

Lessons Learned and Outlook

Plausibel deniability is a practically achievable property, and a feature to desire or to avoid, depending on what you are looking for:

- If you are contributing your personal data to federated learning, plausible deniability implies that there is – information-theoretically – no leakage from the machine learning model encapsulating your sensitive data.
- If you are worried about potential misuse of your data, plausibly denied using this construction, then the data processing entity should commit to a stochastic optimization and publicly known and fixed error metric $\rightarrow$ avoids plausible deniability (using this construction).

Open questions are manifold, such as the link to other security notions, or generalizations.

Appendix

Extension to Vector-Valued Models

Corollary 1.1

Take $k, m, d \geq 1$ and let $f : \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^k$ be parameterized by a vector $\mathbf{p} \in \mathbb{R}^d$, and write f_j for $j = 1, \ldots, k$ to denote the j-th coordinate function. For a fixed parameter vector $\mathbf{p}^*$ and arbitrary training data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathbb{R}^m \times \mathbb{R}^k$, define the error matrix $\mathbf{E}$ row-wise as $\mathbf{E} = (\mathbf{y}_i^\top - f(\mathbf{x}_i, \mathbf{p}^*)^\top)_{i=1}^n \in \mathbb{R}^{n\times k}$. In this matrix, let $\mathbf{e}_j \in \mathbb{R}^n$ be the j-th column.
For all $j = 1, 2, \ldots, k$ and all training points $\mathbf{x}_i$, assume that each $f_j(\mathbf{x}_i, \mathbf{p})$ is totally differentiable w.r.t. $\mathbf{p}$ at (the same point) $\mathbf{p} = \mathbf{p}^*$, with derivative $\mathbf{d}_{i,j} = D_{\mathbf{p}}(f_j(\mathbf{x}_i, \mathbf{p}))(\mathbf{p}^*) \in \mathbb{R}^d$. For each j, define the matrix $\mathbf{M}_j = (\mathbf{d}_{i,j}^\top)_{i=1}^n \in \mathbb{R}^{n\times d}$ and let the rank condition $rank(\mathbf{M}_j|\mathbf{e}_j) > rank(\mathbf{M}_j)$ hold.
Then, there exists a matrix-norm $\|\cdot\|$ on $\mathbb{R}^{n\times k}$ such that $\mathbf{p}^*$ locally minimizes $\|\mathbf{E}(\mathbf{p}^*)\|$, i.e., there is an open neighborhood U of $\mathbf{p}^*$ s.t. $\|\mathbf{E}(\mathbf{p}^*)\| \leq \|\mathbf{E}(\mathbf{p})\|$ for all $\mathbf{p} \in U$.

Corollary 1.2

Under the hypotheses of Theorem 1.1, there is a matrix $\mathbf{C}$ *such that* $\mathbf{p}^*$ *locally minimizes the mean average error* $MAE(\mathbf{C}\cdot\mathbf{e})$ *of the error vector* $\mathbf{e}$*.*

[Boe20] Franziska Boenisch.
Attacks against Machine Learning Privacy (Part 1): Model Inversion Attacks with the IBM-ART Framework, December 2020.

[BSG17] Vincent Bindschaedler, Reza Shokri, and Carl A. Gunter.
Plausible deniability for privacy-preserving data synthesis.
Proceedings of the VLDB Endowment, 10(5):481–492, January 2017.

[CKJQ21] Si Chen, Mostafa Kahla, Ruoxi Jia, and Guo-Jun Qi.
Knowledge-Enriched Distributional Model Inversion Attacks.
arXiv:2010.04092 [cs], August 2021.
arXiv: 2010.04092.

[RKW+21] Stefan Rass, Sandra König, Jasmin Wachter, Manuel Egger, and Manuel Hobisch.
Supervised Machine Learning with Plausible Deniability.
Computers & Security, 112:102506, 2021.

References 2

[ZJP+20] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song.
The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks.
In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 250–258, Seattle, WA, USA, June 2020. IEEE.

[ZZT17] Xi Zhang, Choujun Zhan, and Chi K. Tse.
Modeling the Dynamics of Cascading Failures in Power Systems.
IEEE Journal of Emerging and Selected Topics in Circuits and Systems, 7(2):192–204, 2017.