Privacy Enhancing Technologies
Lecture 13 – Membership Inference Attacks Against Machine Learning Models

Florian Tramèr

AGENDA

1. Recap
2. Deep neural networks
3. Membership inference attacks against neural networks
4. Auditing Private Learning
5. Open problem: white-box membership inference

1 Recap on Membership Inference Attacks

In the last lecture we introduced membership inference attacks, where an adversary tries to infer whether a given data point x was used as input to a mechanism M. We showed that (worst-case) membership inference attacks enjoy a tight connection to differential privacy, and can be used to prove *lower bounds* for DP mechanisms (e.g., the number of counting queries that can be answered with the Gaussian mechanism).

We now turn to membership inference attacks against machine learning models, an area that has seen a lot of study in the past decade. The goal here is for an attacker to infer whether a given data point x was part of the training data of some machine learning model f_θ. In contrast to the prior lecture, the settings we'll consider will typically not be "fully" worst-case. That is, we will aim to measure the privacy leakage of a model trained on some "natural" dataset D (e.g., CIFAR-10 or ImageNet).

2 Deep Neural Networks

A deep neural network $f_\theta : \mathcal{X} \to [0,1]^C$ is a learned function that maps some input data sample $x \in \mathcal{X}$ to an C-class probability distribution; we let $f_\theta(x)_y$ denote the probability of class y.

A (standard feedforward) neural network is a composition of linear transformations and nonlinearities, with the outputs of the last layer passed through a *softmax function* σ to obtain a probability distribution. That is:

$$f(x) = \sigma\Big(\underbrace{f_k \circ \cdots \circ f_1(x)}_{z_\theta(x)}\Big)\,, \tag{1}$$

where each f_i consists of a learned linear transformation followed by a fixed nonlinearity, and $z_\theta : \mathcal{X} \to \mathbb{R}^C$ returns the final *logits* followed by the *softmax*:

$$\sigma(z) = \left[\frac{e^{z_1}}{\sum_i e^{z_i}}, \ldots, \frac{e^{z_C}}{\sum_i e^{z_i}} \right] . \tag{2}$$

Neural networks are trained via gradient descent to minimize some loss function ℓ:

$$\theta_{t+1} \leftarrow \theta_t - \eta \sum_{(x,y) \in B} \nabla_\theta \ell(\theta_t, x, y) \tag{3}$$

Here, B is a batch of random training examples from the training set D, and η is the learning rate, a small constant. For classification tasks, the most common loss function is the cross-entropy loss:

$$\ell(\theta, x, y) = -\log(f_\theta(x)_y) . \tag{4}$$

Overfitting. Since we train machine learning models to minimize the loss function over the training data too well, and have much lower loss on the training data than on the general population (i.e., the empirical risk $\widehat{\mathcal{L}}_D(\theta)$ is much lower than the population risk $\mathcal{L}(\theta)$). This is typically referred to as *overfitting*.

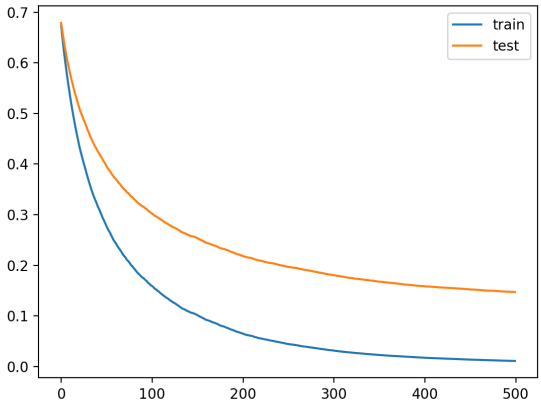

Figure 1: Deep neural networks tend to overfit to the training data. The training loss (in blue) typically decreases faster than the test loss (in orange).

Overfitting relates to privacy in that it reveals whether the trained model has *memorized* some features of the training data that do not generalize to the general population. But some presence of overfitting does not necessarily imply that individual data points are at risk of being leaked. All it tells us is that the model has memorized some aspects of the training data *on average*.

3 Membership Inference Attacks Against Neural Networks

Suppose we play the membership inference game where the mechanism M trains and returns a neural network model f_θ on some training data D. What strategies could the adversary use to infer whether the model was trained on D_1 or D_0 (i.e., whether x was in the training data or not)?

One option could be to look at the learned parameters θ and see if they somehow encode some information about x. We'll get back to this idea later (it's in fact surprisingly hard to do this). Instead, we could just look at the model's *output* on x—in particular, the model's loss $\ell(\theta, x, y)$. This is called as *black-box* attack because we only need query access to the trained model.

The intuition behind using the model's loss as a membership indicator is that we expect the loss to be lower for data points in the training data, on average, due to overfitting:

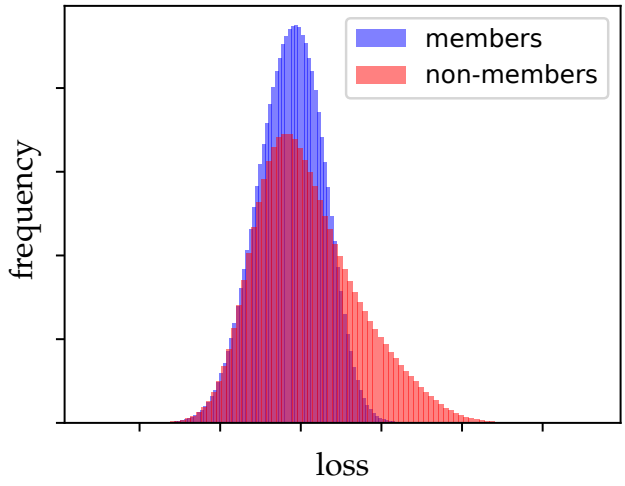

Figure 2: Histogram of the loss of a neural network trained on CIFAR-10, for samples from the training set (in red) and from the test set (in blue).

3.1 A Simple Attack: Global Loss Thresholding

The earliest membership inference attacks against machine learning models were directly inspired by the (average-case) overfitting phenomenon shown above [SSSS17, YGFJ18]. Since $\widehat{\mathcal{L}}_D(\theta) < \mathcal{L}(\theta)$, we can set some threshold τ on the loss ℓ and declare any point x to be a member if its loss is below the threshold:

$$\mathcal{A}_{\text{global}}(\theta, x, y) = \begin{cases} 1 & \text{if } \ell(\theta, x, y) \leq \tau \\ 0 & \text{otherwise} \end{cases}$$

However, it turns out that this is not a very good attack. For example, if we look at the histogram of losses above in Figure 2, we can see that there is no threshold that would let us confidently infer that any sample is a member.[1]

But this is when we consider the attack's performance *on average* across all samples. What we'd rather want is an attack tailored to each individual sample that we want to attack. But here, a global threshold won't do the trick.

3.2 The Need for Difficulty Calibration

The reason that a global thresholds on the loss gives a poor attack is that not all samples are equally hard to learn. This is illustrated in Figure 3 below. We train a large number of models on random subsets of CIFAR-10, and measure the loss of each model on four different points. For each of the four images, we then plot the loss values for models that were trained on that image (in red) and models that were not trained on that image (in blue).

[1] Note that we can however confidently infer that the sample is *not* a member if its loss is high enough.

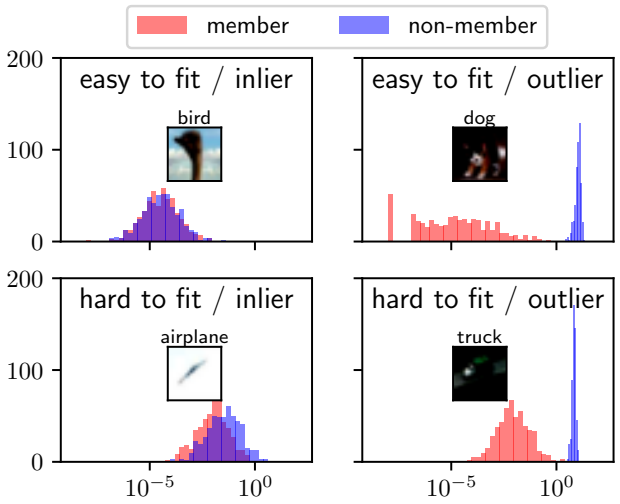

Figure 3: Not all samples are equally hard (figure copied from [CCN+22]). For four samples from CIFAR-10, we plot a histogram of a model's loss on that sample when it was in the training set (in red) and when it was not (in blue). Only outliers are easy to infer membership for. The losses of different samples can also be on very different scales, depending on how hard they are to learn.

These images were chosen to showcase some extreme behaviors:

- The bird on the top left is a prototypical "inlier" (i.e., a point that is typical of the training data; there are many similar birds in the CIFAR-10 dataset). For such points, the model generalizes well, and so the loss is essentially the same whether the model was trained on this specific image or not.
- However, some inliers are still "harder" to learn than others. The airplane on the bottom left is also an inlier (i.e., it's presence in the training data has little influence on the model's loss), but it belongs to a category of images that are harder to learn than the top-left bird.
- The dog on the top right is an outlier (i.e., other dogs in the dataset are typically not as dark). Models that are not trained on this image tend to misclassify it and get very high loss. But when a model is trained on this image, it usually memorizes its label and gets low loss.
- The truck on the bottom right is also an outlier, but this one is harder for the model to memorize; even if the model was trained on the truck, its loss is still rather high.

From the figure, it's clear that a single global threshold τ will not work well in distinguishing members from non-members for all data points.

3.3 A Stronger Attack: Per-Sample Likelihood Ratios

A better attack strategy would be to set a different threshold on the loss ℓ for each data point x. But still, this would be somehow crude because there is no way for the attack to express its uncertainty over whether a point is a member or not. For example, if you set a threshold of τ for the loss ℓ of some point x, and you observe a loss of 1.01τ, you should intuitively be more uncertain about whether x is a member or not than if you observed a loss of 100τ.

This leads us to the **optimal** way of formulating the membership inference attack as a *likeli-*

hood ratio test. Define the *null hypothesis* H_0 that x is not a member of the training data, and the *alternative hypothesis* H_1 that it is a member. Further let $\texttt{View} = M(D_b)$ be the output of the mechanism that the adversary gets to see. The likelihood ratio test then compares the ratio of the likelihoods of the null and alternative hypotheses, given the observed data

$$\Lambda(\texttt{View}) = \frac{\Pr[\texttt{View} \mid H_0]}{\Pr[\texttt{View} \mid H_1]}, \tag{5}$$

and rejects the null hypothesis if $\Lambda(\texttt{View}) \geq \tau$ for some threshold τ.

The Neyman-Pearson lemma tells us that this likelihood ratio test is the most powerful test for distinguishing between the null and alternative hypotheses, at any given false positive rate [NP33]!

Now it remains to actually compute the likelihoods $\Pr[\texttt{View} \mid H_0]$ and $\Pr[\texttt{View} \mid H_1]$. We don't know of any way to do this in closed form, since the probability here is taken over the entire process of training a deep neural network. Instead, we could aim to estimate the likelihoods empirically. We describe here the approach of [CCN+22], but there are others with slightly different assumptions (e.g., [YMM+22]).

First, let's again consider that we are in a black-box setting where the adversary just observes the loss for some target x and some label y, i.e., $\texttt{View} = \ell(\theta, x, y)$. So we now want to estimate

$$\Pr[\ell(\theta, x, y) \mid H_b] \tag{6}$$

To do this, [CCN+22] suggest to train multiple "IN" and "OUT" models that mimic the membership inference game. That is, the attacker samples some training data D_{atk} from the distribution $\mathcal{P}$ and then trains a model $f_{\theta^{\text{IN}}}$ on $D_{atk} \cup \{x\}$ and a model $f_{\theta^{\text{OUT}}}$ on D_{atk}. After repeating this process N times (for the same target x), we get $2N$ loss samples:

$$\text{Loss}_{\text{IN}} = \left\{ \ell(\theta^{\text{IN}}, x, y) \;\middle|\; \begin{array}{c} D_{atk} \sim \mathcal{P} \\ \theta^{\text{IN}} \leftarrow M(D_{atk} \cup \{x\}) \end{array} \right\}$$

$$\text{Loss}_{\text{OUT}} = \left\{ \ell(\theta^{\text{OUT}}, x, y) \;\middle|\; \begin{array}{c} D_{atk} \sim \mathcal{P} \\ \theta^{\text{OUT}} \leftarrow M(D_{atk}) \end{array} \right\}$$

These loss values are what we previously showed in Figure 3. The final step in the attack from [CCN+22] is to fit a Gaussian distribution to both of these sets of loss samples, so that we can estimate the probability in eq. (6) even when it is very small (and thus estimate the attack's TPR for very small FPR). Then, given these distributions $\mathcal{N}(\mu_{\text{IN}}, \sigma^2_{\text{IN}})$ and $\mathcal{N}(\mu_{\text{OUT}}, \sigma^2_{\text{OUT}})$, the final likelihood ratio test is given by

$$\Lambda(\ell) = \frac{\Pr[\ell \mid H_0]}{\Pr[\ell \mid H_1]} = \frac{\mathcal{N}(\mu_{\text{IN}}, \sigma^2_{\text{IN}})(\ell)}{\mathcal{N}(\mu_{\text{OUT}}, \sigma^2_{\text{OUT}})(\ell)} \tag{7}$$

The performance of this attack on CIFAR-10 is shown in Figure 4. Here, we show the attack's TPR at a fixed FPR of 0.1%, for each of the 50'000 samples in CIFAR-10. We can see that a small number of samples are incredibly vulnerable to membership inference, with the attack reaching a TPR of almost 100% for some samples. The majority of samples, however, are very hard to infer membership for. The attack achieves at best a few percent of TPR for most samples.

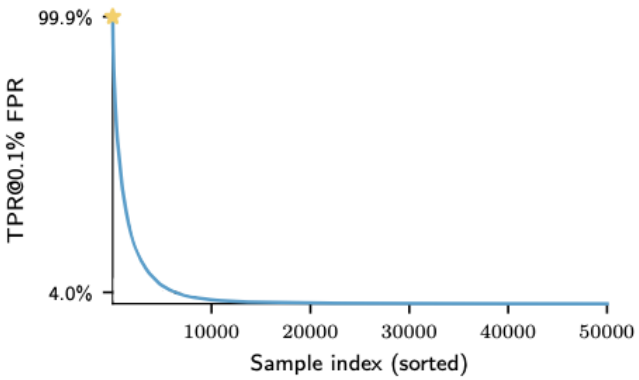

Figure 4: Performance of the Likelihood Ratio Attack (LiRA) of [CCN+22] (figure copied from [AZT24]). For each of the 50'000 samples in CIFAR-10, we plot the attack's TPR at a fixed FPR of 0.1% (sorted by TPR).

This is illustrative of how privacy is a *worst-case* notion: while most of the data samples might be safe, a successful privacy enhancing technology should protect *all* data samples from attack, even samples that are highly unusual (maybe those are the ones that are most important to protect...).

Training a model with differential privacy (e.g., with DP-SGD) would indeed prevent membership inference attacks for *all* data points, if the privacy budget ε is set appropriately.

4 Auditing Private Learning with Membership Inference

Recall Corollary 1 from the last lecture, which we restate here for convenience:

Corollary 1. A mechanism M is (ε, δ)-DP *if and only if* for all adversaries in the membership inference game,

$$e^{\varepsilon} \geq \max\left(\frac{\mathrm{TNR} - \delta}{\mathrm{FNR}}, \frac{\mathrm{TPR} - \delta}{\mathrm{FPR}}\right)$$

One way we can use this result is to select parameters (ε, δ) so that we get formal guarantees on the performance of *any* membership inference attack. For example, if we want to ensure that no attack can reach a TPR of 99% at a FPR below 1%, we should set $\varepsilon \leq \ln 99 \approx 4.59$.

But we can also apply this result in the other direction: given some empirical evaluation of a membership inference attack, we can use it to lower-bound the privacy parameters (ε, δ) that a mechanism satisfies. There are many applications of this in the literature, which we discuss below.

4.1 Debugging DP-SGD Implementations

As with any security mechanism, private training algorithms such as DP-SGD might not actually provide the formal guarantees given by the privacy analysis. This could be due to bugs in the implementation, or due to flaws in the privacy analysis itself. That is, even if we have a purported proof that our DP-SGD algorithm satisfies, say $(1, 10^{-8})$-DP, it is possible

that the actual privacy loss is much higher in practice.

Membership inference attacks are a way to empirically evaluate the privacy of a mechanism. If a mechanism claims ε privacy, and yet we can build a membership inference attack that achieves $\mathrm{TPR}/\mathrm{FPR} \geq e^{\varepsilon}$, then we have evidence that there is a bug somewhere.

As an example, in [TTS+22] we looked at a proposed variant of DP-SGD that claimed very strong privacy guarantees. We then ran this mechanism M many times on two datasets D and $D \cup \{x\}$, and estimated the TPR and FPR of the LiRA attack of [CCN+22]. While the mechanism claimed $\varepsilon \approx 0.2$ privacy, we found that the actual privacy budget was at least $\varepsilon' > 2.7$. This was due to a bug in the implementation of the mechanism, which caused the added noise to be divided by the batch size m.

4.2 Understanding the Tightness of the DP-SGD Analysis

Recall the privacy analysis of the DP-SGD algorithm which we sketched in lecture 11. This analysis is known to be tight in a somewhat contrived worst-case setting, where the following conditions are met:

1. All examples in D have gradients that are zero (i.e, $g_i = \vec{0}$) while the target sample x has a (clipped) gradient of the form $g(x) = [C, 0, \ldots, 0]$. (with slightly more generality, we can also make the weaker assumption that all samples have gradients orthogonal to the gradient of x).
2. The attacker gets to observe all intermediate learning steps $\theta_0, \ldots, \theta_T$.

It is an open question whether these conditions are necessary for the DP-SGD analysis to be tight. The latter assumption, in particular, is quite unrealistic: in practice, the attacker would typically only get to see the final trained model θ_T, and not any intermediate steps.

Once we relax these assumptions, our best membership inference attacks yield lower-bounds on the privacy parameter ε that don't match the upper-bounds given by the DP-SGD analysis [JUO20, NST+21]. And so this begs the question: are our attacks not strong enough, or is the DP-SGD analysis too conservative in these settings?

This is an active area of research. It is known that in convex settings, releasing only the final model θ_T can provide better privacy guarantees than releasing all intermediate steps [FMTT18]. But it remains unclear whether a similar result holds for non-convex settings, such as training deep neural networks.

4.3 Empirical Evaluation of Heuristic Privacy Defenses

Since the DP-SGD analysis might be overly pessimistic, some researchers have also proposed heuristic privacy defenses for training deep neural networks. These defenses do not have any formal privacy guarantees, and so the only way to evaluate them is via empirical membership inference attacks.

This can be tricky though. In recent work, we showed that a number of heuristic defenses against membership inference attacks are in fact *not* effective when evaluated properly [AZT24].

5 Open Problem: White-box Membership Inference

The attacks we have discussed so far are all *black-box* attacks, since the adversary only relies on observing the model's output on some target point x, and not any of the actual learned parameters θ.

Intuitively, we would expect a *white-box* attack to be much stronger, since the adversary has access to a lot more information about the learned model. Some works leveraged this additional information to build attacks that beat the naive global loss thresholding attack from Section 3.1 [NSH19].

Yet, we now know that these attacks were not particularly strong to begin with, and that per-sample likelihood ratio attacks from Section 3.3 are in fact a lot stronger. It is an open question whether white-box access to the model's parameters could help us build a stronger likelihood ratio test. The idea here would be to estimate the probabilities

$$\Pr[\theta \mid H_b] \tag{8}$$

but we don't know how to do this for multiple reasons. First, the parameters θ are high-dimensional, and so empirical estimation might require training millions of models. And even if we did that, the problem is that every time we train a new model, the ordering of the parameters might completely change (neural networks contain a number of *symmetries*, e.g., permuting all weights in one layer and applying the inverse permutation to the next layer yields an equivalent model). And so even if we had millions of samples of the parameters θ, for models trained both with and without the target sample x, it is not obvious how to use these to estimate the likelihoods in eq. (8).

If you have any ideas about how to approach this problem, please let me know! This is a fascinating open research problem, and thus a great place to end this course. As mentioned in the first lecture, this might be your *first* and *last* class on privacy enhancing technologies, so now's the time to go out there and apply what you learned! I hope you enjoyed the ride and got a flavor of the amazing results and open research questions in this area.

References

[AZT24] Michael Aerni, Jie Zhang, and Florian Tramèr. Evaluations of machine learning privacy defenses are misleading. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024.

[CCN+22] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[FMTT18] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[JUO20] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private SGD? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[NP33] Jerzy Neyman and Egon Sharpe Pearson. IX. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of*

London. Series A, Containing Papers of a Mathematical or Physical Character, 231(694-706):289–337, 1933.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[NST+21] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[TTS+22] Florian Tramer, Andreas Terzis, Thomas Steinke, Shuang Song, Matthew Jagielski, and Nicholas Carlini. Debugging differential privacy: A case study for privacy auditing. *arXiv preprint arXiv:2202.12219*, 2022.

[YGFJ18] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[YMM+22] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3093–3106, 2022.