Machine Learning Models That Remember Too Much

BY

BHAVANAREDDY MANDADI(27096467)
Outline

• Overview
• Machine Learning Systems
• How Non-ML-experts Train ML models
• Behind the Scene: ML Pipeline
• Isolated Training Environment
• After Training Attacks
• Countermeasures
• Takeaways
• Paper Referred
Overview:

A new threat model in ML:

1. Non-expert client train ML model with adversary provided code.
2. Training runs in an isolated environment to prevent leakage.
3. Adversary gain access to resulting model.
Overview:

Attacks in this threat model:

• Let model learn two tasks at the same time
Overview:

**Attacks in this threat model:**

- Let model learn two tasks at the same time
- When model is available, adversary can extract
Overview:

**Attacks in this threat model:**

- Let model learn two tasks at the same time
- When model is available, adversary can extract
Machine Learning Systems

- **Training data** $D_{train} = \{(x_i, y_i)\}_{i=1}^n$
  - feature vector (e.g. image)
  - class label (e.g. identity)

- **ML model** $f_\theta(x) = y$ for prediction.
  - a set of parameters (e.g. weight matrices)

- Deploy the model for application or publish the model for others to use.
How Non-ML-experts Train ML models:

Non-expert clients wish to perform ML task in their dataset:

• Use third-party easy-to-use ML library

• Use “ML-as-a-Service” platform
Behind the Scene: ML Pipeline
Behind the Scene: ML Pipeline
Behind the Scene: ML Pipeline

- Regularization term penalizes $\theta$ if too complex
- Regularization term penalizes $\theta$ if too complex
- Training algorithm defines model and finds optimal $\theta$
- $\Omega$: Regularization term penalizes $\theta$ if too complex
- $\Omega$: Regularization term penalizes $\theta$ if too complex
Behind the Scene: ML Pipeline

- regularization term penalizes $\theta$ if too complex
- regularization term penalizes $\theta$ if too complex
- **training algorithm defines model and finds optimal $\theta$**
- $\Omega$: regularization term penalizes $\Omega$ if too complex
Behind the Scene: ML Pipeline

Clients’ data can be sensitive
Black-box code is untrusted
Make the black-box SECURE!

Potential Solution:
Isolated training environment

- $\lambda$: regularization parameter
- $\lambda$: regularization parameter
- $\text{training algorithm}$
- $\Omega$: regularization term penalizes $\theta$ if too complex
Isolated Training Environment:

Preserve data privacy when running untrusted training code

In academia:
- Software-based: CQSTR [Zhai et al 2016]

In practice:
- Third-party provide training code.
- Code can be close-sourced.
- Can cut off network access to isolate training!

 Warns about potential leakage

Many of the algorithms in the platform require accessing the Internet. Some algorithms began sending data outside of the Algorithmia platform during execution. If an algorithm's permissions indicate that it requires internet access, please be aware that there is the potential for your data to leave the Algorithmia platform.
After Training:

Training environment is secure. What happens after training?

- Client makes the model available if performance is good
- For example: deploy the model for application and allow others to query

After training, data is gone and model can be accessed by others.
Threat Model:

1. Client train model with **adversary provided training code in secure and isolated environment**.

2. After model passes validation, **adversary** gains access to the model.

What attacks arise in this threat model?
Attacks:

- Client makes model available **solely** based on **test performance**... and may not even ask: What else did the model capture about my training data?

- **Provide training code** to learn a model that does **two tasks simultaneously**!
Attacks:

- White-box Attacks leak through parameters
  - LSB Encoding
  - Correlated Value Encoding
  - Sign Encoding

- Black-box Attack leak through predictions
  - Capacity Abuse Attack

This talk focuses on attacks against deep learning models.

Adversary can only send queries.
White-box Attacks

**LSB encoding**: write data into the LSB of benignly trained theta.

- Simple but effective.

**More natural way: encode secrets in theta during training**

- Express malicious task as a regularization term!

- Solver finds theta that optimizes two tasks at the same time!
White-box Attacks:

**Standard ML use regularization term to constrain \( \theta \)**

- L-1 regularization: \( \Omega(\theta) = \lambda \sum_i |\theta_i| \)
- L-2 regularization: \( \Omega(\theta) = \lambda \sum_i \theta_i^2 \)

**Engineered “regularization” term to bias \( \theta \) to encode \( s \)**

\[
\Omega_{mal}(\theta, s) = -\lambda_c \frac{\sum_i |(\theta_i - \bar{\theta})(s_i - \bar{s})|}{\text{std}(\theta) \text{std}(s)}
\]

Correlated Value Encoding:
- Correlated value can leak data
- Extract \( s \) as a sequence of real numbers
- Maximize the correlation between \( \theta \) and \( s \)

Sign Encoding:
- Sign can leak one bit
- Extract \( s \) as bitstring made of \{-1, +1\}
- Match signs of \( \theta \) with \( s \)
Decode from White-box Model:

- **Correlation Decoding**: apply linear transformation to the parameters
- **Sign Decoding**: read the signs and interpret them as bitstring
White-box Attacks Results:

1. Test Performance (compare to baseline benignly trained model)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Test acc ±δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>91.09 – 1.80</td>
</tr>
<tr>
<td>LFW</td>
<td>87.91 – 0.08</td>
</tr>
<tr>
<td>FaceScrub(G)</td>
<td>97.27 – 0.16</td>
</tr>
</tbody>
</table>

2. Encoding Quality
Black-box Attack:

• So, White-box attacks require fully disclosed models. In this client might only provide prediction API.

• So, we really need a way to leak data through predictions which is done by: **Capacity Abuse Attack**. This attack is motivated from a recent finding in deep learning which basically says deep learning models can memorize the randomly labeled data perfectly well.
Capacity Abuse Attack:

Data Augmentation: widely used in image tasks
- Example: rotation, scaling, distortion.
- Help model generalizes better!

Malicious Data Augmentation:
- Encode secrets in the labels!
- Label can leak one bit (for binary classification).
- Abuse model’s capacity by memorizing $D_{mat}$
Malicious Data Augmentation:
Black-box Attack Results:

1. Test Performance (compare to baseline benignly trained model)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data added</th>
<th>Test acc ±δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>49,000</td>
<td>92.21 ± 0.69</td>
</tr>
<tr>
<td>LFW</td>
<td>58,000</td>
<td>88.17 ± 0.34</td>
</tr>
<tr>
<td>FaceScrub(G)</td>
<td>170,000</td>
<td>96.94 ± 0.50</td>
</tr>
</tbody>
</table>

δ: difference to baseline

2. Encoding Quality

Ground Truth

Reconstructed
Summary (Learn Two Tasks Simultaneously):

- **Primary Task**: Good test performance
- **Malicious Task**: Leak the training data

**Standard objective function**:
\[
\Omega(\theta) + \frac{1}{n} \sum_{i} L(y_i, f_\theta(x_i))
\]

**Engineered term**:
\[
\Omega_{mal}(\theta, s)
\]

**ML models can do both tasks well!**

**White-box Attacks**

**Black-box Attack**

**Client’s training data**:
\[ y_1, y_2, \ldots, y_n \]

**Data with secrets in labels**:
\[ b_1, b_2, \ldots, b_l \]
Countermeasures:

Check the difference in parameter distributions?

Correlated value, Sign, and Capacity abuse histograms showing distributions for benign and correlated states.

Need to understand what is the “normal” parameter distribution.
Take-aways:

• We have seen if training code is untrusted even isolated systems cannot prevent leakage.

• Models do well on primary task... and learn other things!

• Thus, ML code should not be applied blindly to sensitive dataset and basically we should understand the code, ML model should only do what it is asked to do!
Paper Referred:

Paper Title: Machine Learning Models that Remember Too Much
Published On: October 31, 2017 at ACM CSS
Paper By: Congzheng Song, Thomas Ristenpart, Vitlay Shmatikov
THANK YOU!