Machine Unlearning of Features and Labels

Alexander Warnecke\textsuperscript{1}, Lukas Pirch\textsuperscript{1}, Christian Wressnegger\textsuperscript{2}, Konrad Rieck\textsuperscript{1}

\textsuperscript{1}Technische Universität Berlin
\textsuperscript{2}Karlsruhe Institute of Technology
Machine Learning

Data points

Features

Labels

Training
Machine Unlearning

- Algorithms to remove information from ML models
  - Necessary to fulfill privacy policies like GDPR or CCPA
  - So far, removal of entire datapoints
Machine Unlearning

- Algorithms to remove information from ML models
  - Necessary to fulfill privacy policies like GDPR or CCPA
  - So far, removal of entire data points
- We extend the concept of Unlearning to Features and Labels
Approach

▶ Input given by model and its parameters $\theta^*$
▶ Framework for unlearning: $\theta = \theta^* + U(Z, \tilde{Z})$
  ▶ $Z$ contains the datapoints to be fixed, $z = (x, y)$
  ▶ $\tilde{Z}$ contains the corrected datapoints $\tilde{z} = (x + \delta_x, y + \delta_y)$
Approach

- Input given by model and its parameters $\theta^*$
- Framework for unlearning: $\theta = \theta^* + U(Z, \tilde{Z})$
  - $Z$ contains the datapoints to be fixed, $z = (x, y)$
  - $\tilde{Z}$ contains the corrected datapoints $\tilde{z} = (x + \delta_x, y + \delta_y)$
- Difference in gradients of loss used as basis

$$\Delta(Z, \tilde{Z}) = \sum_{\tilde{z} \in \tilde{Z}} \ell(\tilde{z}, \theta^*) - \sum_{z \in Z} \nabla \ell(z, \theta^*)$$
Approach

- Input given by model and its parameters $\theta^*$
- Framework for unlearning: $\theta = \theta^* + U(Z, \tilde{Z})$
  - $Z$ contains the datapoints to be fixed, $z = (x, y)$
  - $\tilde{Z}$ contains the corrected datapoints $\tilde{z} = (x + \delta_x, y + \delta_y)$
- Difference in gradients of loss used as basis
  \[
  \Delta(Z, \tilde{Z}) = \sum_{\tilde{z} \in \tilde{Z}} \ell(\tilde{z}, \theta^*) - \sum_{z \in Z} \nabla \ell(z, \theta^*)
  \]
- $U(Z, \tilde{Z}) = -\tau \Delta(Z, \tilde{Z})$ (First-Order)
Approach

- Input given by model and its parameters $\theta^*$
- Framework for unlearning: $\theta = \theta^* + U(Z, \tilde{Z})$
  - $Z$ contains the datapoints to be fixed, $z = (x, y)$
  - $\tilde{Z}$ contains the corrected datapoints $\tilde{z} = (x + \delta_x, y + \delta_y)$
- Difference in gradients of loss used as basis

$$\Delta(Z, \tilde{Z}) = \sum_{\tilde{z} \in \tilde{Z}} \ell(\tilde{z}, \theta^*) - \sum_{z \in Z} \nabla \ell(z, \theta^*)$$

- $U(Z, \tilde{Z}) = -\tau \Delta(Z, \tilde{Z})$ (First-Order)
- $U(Z, \tilde{Z}) = -H_{\theta^*}^{-1} \Delta(Z, \tilde{Z})$ (Second-Order)
Certified Unlearning

- How can we guarantee that information has been removed?

- Add random noise to parameters
- Bound the difference between retraining and unlearning
  \[ e - \epsilon \leq P_{\text{model after unlearning}} \]
  \[ P_{\text{retrained model}} \leq e + \epsilon \]

- Inspired by the concept of differential privacy (DP)
- Theorem
- Both update strategies are certified for convex loss functions with bounded derivatives.
Certified Unlearning

- How can we guarantee that information has been removed?
- Guarantee that unlearning is indistinguishable from retraining
  - Add random noise to parameters
  - Bound the difference between retraining and unlearning

\[ e^{-\epsilon} \leq \frac{P(\text{"Model after unlearning"})}{P(\text{"Retrained model"})} \leq e^{\epsilon} \]
Certified Unlearning

- How can we guarantee that information has been removed?
- Guarantee that unlearning is indistinguishable from retraining
  - Add random noise to parameters
  - Bound the difference between retraining and unlearning

\[ e^{-\epsilon} \leq \frac{P(\text{"Model after unlearning"})}{P(\text{"Retrained model"})} \leq e^\epsilon \]

- Inspired by the concept of differential privacy (DP)
Certified Unlearning

- How can we guarantee that information has been removed?
- Guarantee that unlearning is indistinguishable from retraining
  - Add random noise to parameters
  - Bound the difference between retraining and unlearning

\[
e^{-\epsilon} \leq \frac{P(\text{“Model after unlearning”})}{P(\text{“Retrained model”})} \leq e^\epsilon
\]

- Inspired by the concept of differential privacy (DP)
- Theorem
  - Both update strategies are certified for convex loss functions with bounded derivatives.
Evaluating Unlearning

▶ We propose three criteria for evaluation:

- **Efficacy**: We require measures that information has been removed.
- **Fidelity**: Classification performance should be close to the original model.
- **Efficiency**: The Unlearning algorithm must be faster than retraining.

All criteria must hold at the same time! We don’t need fast algorithms with low fidelity or efficacy, or algorithms with high fidelity or efficacy that are slow.
Evaluating Unlearning

- We propose three criteria for evaluation
  - Efficacy
    - We require measure that information has been removed
  - Fidelity
    - Classification performance should be close to the original model
  - Efficiency
    - The Unlearning algorithm must be faster than retraining

All criteria must hold at the same time! We don't need fast algorithms with low fidelity or efficacy or algorithms with high fidelity or efficacy that are slow.
Evaluating Unlearning

- We propose three criteria for evaluation
  - Efficacy
    - We require measure that information has been removed
  - Fidelity
    - Classification performance should be close to the original model
Evaluating Unlearning

- We propose three criteria for evaluation
  - Efficacy
    - We require a measure that information has been removed
  - Fidelity
    - Classification performance should be close to the original model
  - Efficiency
    - The Unlearning algorithm must be faster than retraining
Evaluating Unlearning

- We propose three criteria for evaluation
  - Efficacy
    - We require a measure that information has been removed
  - Fidelity
    - Classification performance should be close to the original model
  - Efficiency
    - The Unlearning algorithm must be faster than retraining
- All criteria must hold at the same time! We don’t need
  - Fast algorithms with low fidelity or efficacy
  - Algorithms with high fidelity or efficacy that are slow
Case Study: Generative Language Models

Learning Model
- Character based language model based on LSTM
- Trained on the novel, "Alice in wonderland"
- Insertion of a canary sentence to induce memorization
  - "'My telephone number is 0123456789', said Alice."

---

1 The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, Usenix Security, 2019
Case Study: Generative Language Models

Learning Model
- Character based language model based on LSTM
- Trained on the novel "Alice in wonderland"
- Insertion of a canary sentence to induce memorization\(^1\)
  
  
  "’My telephone number is 0123456789’, said Alice."

Task
- Unlearn the memorized number by changing features and labels
  
  
  "’My telephone number is not here’, said Alice."\(^\prime\)

\(^1\)The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, Usenix Security, 2019
Case Study: Generative Language Models

- **Learning Model**
  - Character based language model based on LSTM
  - Trained on the novel "Alice in wonderland"
  - Insertion of a canary sentence to induce memorization\(^1\)
  - "’My telephone number is 0123456789’, said Alice."

- **Task**
  - Unlearn the memorized number by changing features and labels
  - "’My telephone number is not here ’, said Alice."

- **Evaluation**
  - Exposure metric for efficacy of unlearning
  - Accuracy on training data for fidelity

---
\(^1\)The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, Usenix Security, 2019
Start sentence "My telephone number is "
Induces probability distribution over $36^{10}$ possible completions
Start sentence "My telephone number is "
- Induces probability distribution over $36^{10}$ possible completions
- To measure how surprised the model is we use log-perplexity

$$P_x(x_1 \ldots x_{10}) = - \log(Pr(x_1 \ldots x_{10}))$$
Start sentence "My telephone number is "
Induces probability distribution over $36^{10}$ possible completions.
To measure how surprised the model is we use log-perplexity

$$\Pr(x_1 \ldots x_{10}) = -\log(\Pr(x_1 \ldots x_{10}))$$

Sample $10^7$ random completions to approximate distribution
Unlearning unintended memorization - Efficacy

- Start sentence "'My telephone number is "
  - Induces probability distribution over $36^{10}$ possible completions,
  - To measure how surprised the model is we use log-perplexity

$$P_x(x_1 \ldots x_{10}) = -\log(P_r(x_1 \ldots x_{10}))$$

- Sample $10^{10}$ random completions to approximate distribution
- "'My telephone number is 8584881081"

![Skewnorm Fit](image)
Start sentence "My telephone number is "
- Induces probability distribution over $36^{10}$ possible completions.
- To measure how surprised the model is we use log-perplexity

$$P_x(x_1 \ldots x_{10}) = - \log(P(x_1 \ldots x_{10}))$$

- Sample $10^{10}$ random completions to approximate distribution
- "My telephone number is 0123456789"

![Graph showing log-perplexity and relative frequency](image)
Unlearning unintended memorization - Efficacy

- Start sentence "My telephone number is"
  - Induces probability distribution over $36^{10}$ possible completions
  - To measure how surprised the model is we use log-perplexity
  
  $$P_x(x_1 \ldots x_{10}) = -\log(Pr(x_1 \ldots x_{10}))$$

- Sample $10^{10}$ random completions to approximate distribution
- "My telephone number is 0123456789"

![Graph showing log-perplexity distribution before and after unlearning with a Skewnorm fit.]

Relative Frequency

Log-Perplexity

50 100 150 200 250

0.00 0.01 0.02

Before Unlearning After Unlearning Skewnorm Fit
Unlearning unintended memorization - Efficacy

Result

Removing unintended memorization is surprisingly simple and renders extraction of memorized information infeasible.

![Graph showing log-perplexity before and after unlearning]
Unlearning unintended memorization - Fidelity & Efficiency

- Performance is close to retraining for small number of canaries
- Substantial speedup compared to retraining (up to $100 \times$)

![Graph showing accuracy and time for different methods.

- First Order
- Second Order
- Fine-tuning
- SISA

Accuracy vs. # Affected samples

Time (s) vs. # Affected samples
Unlearning unintended memorization

- How is the canary completed after unlearning?
  - Prediction of replacement?
  - Gibberish caused by unlearning?
Unlearning unintended memorization

How is the canary completed after unlearning?

Completions preserve structure of the dataset and punctuation

| Length | Replacement         | My telephone number is ... |
|--------|---------------------|---------------------------|
| 5      | taken               | ‘... mad!’ ‘prizes! said the lory confused . . . |
| 10     | not there_          | ‘... it,’ said alice. ‘that’s the beginning . . . |
| 15     | under the mouse     | ‘... the book!’ she thought to herself ‘the . . . |
| 20     | the capital of paris| ‘... it all about a gryphon all the three of . . . |
Case Study: Poisoning Attacks

- **Model**
  - Convolutional network (VGG) for image classification (CIFAR-10)
  - Flipping of image labels to reduce performance
Case Study: Poisoning Attacks

- **Model**
  - Convolutional network (VGG) for image classification (CIFAR-10)
  - Flipping of image labels to reduce performance

- **Task**
  - Unlearn the poisoned samples by correcting the labels

---

**Evaluation**

- Accuracy on test data after unlearning for Efficacy & Fidelity
Case Study: Poisoning Attacks

▶ **Model**
  ▶ Convolutional network (VGG) for image classification (CIFAR-10)
  ▶ Flipping of image labels to reduce performance

▶ **Task**
  ▶ Unlearn the poisoned samples by correcting the labels

▶ **Evaluation**
  ▶ Accuracy on test data after unlearning for Efficacy & Fidelity
Unlearning Poisoning

- No approach can remove poisoning effect completely
- Great speedup compared to retraining

![Accuracy and Time Graphs](image)
Limitations

- **Size of changes matters**
  - Our approach can fix defects caused by few erroneous samples
  - Retraining is inevitable at some point

- **Certification only for convex loss functions**
  - Modern neural networks have usually non-convex loss
  - Could be mitigated by application to final layers only

- **Unlearning requires detection**
  - Finding data to be removed is a hard problem in the real world
Conclusion

▶ We propose two unlearning updates $\theta = \theta^* + \mathcal{U}(Z, \tilde{Z})$
  ▶ First order update uses gradient information
  ▶ Second order update includes Hessian matrix
▶ We derive conditions to enable certified unlearning
▶ We show that our approach can solve security problems