Student Name: Mohammad Mohammad Beigi - Pantea Amoie
Student ID: 99102189 - 400101656
Subject: Privacy in Machine Learning

Introduction To Machine Learning - Dr. R. Amiri
Final Project

Introduction

Introduction

This project emphasizes the importance of privacy in machine learning and introduces the concepts that will be explored, including Machine Unlearning ([1]). and training private models. The project aims to address how to ensure data privacy while using powerful machine learning models.

1. Machine Unlearning

Machine unlearning refers to a model's ability to forget specific data points without retraining from scratch. This capability is crucial for:

- **Privacy and Data Protection:** Complying with regulations like GDPR.
- **Error Correction:** Correcting mistakes in the training data.
- **Data Management:** Keeping models relevant by removing outdated information.

SISA Algorithm

The SISA (Sharded, Isolated, Sliced, and Aggregated) algorithm is introduced as an efficient method for machine unlearning:

1. **Sharding:** Dividing the training data into smaller subsets (shards).
2. **Isolation:** Training separate models for each shard.
3. **Slicing:** Further dividing each shard into slices and training sequentially.
4. **Aggregation:** Combining the outputs of all shards to form the final model.

The algorithm facilitates selective data removal by targeting specific shards and slices, minimizing computational cost compared to full retraining.

Questions

Some questions which may arise for this section are:

1. **Aggregation Methods:** Name 3 aggregation methods you would propose for this algorithm. To your best knowledge, reason each method's strengths and weaknesses.

   Answer:

1. Majority Voting

Description: Each shard's model provides a prediction, and the final output is determined by the majority vote among these predictions.

Strengths:

- Simple to implement and understand. computationally efficient.
- Robust to individual model errors, as the majority decision can mitigate the impact of outliers.
- Works well when the individual models are relatively independent and diverse.

Weaknesses:

- Loses information about the runner-up classes, potentially reducing accuracy in cases where constituent models assign high scores to multiple classes.
- Does not consider the confidence levels of individual predictions, potentially leading to suboptimal decisions if most models are uncertain.
- May not perform well if the models are not diverse or if there is a strong bias in the training data of individual shards.
- Can be computationally expensive for a large number of shards.

2. Averaging Prediction Vectors

Description: Each model trained on a shard produces a prediction vector. This vector contains probabilities for each class, representing the model's confidence in each possible output. Instead of simply taking the majority vote of the final predictions from each model, averaging prediction vectors involves combining these probability vectors from all the shards. The combined (averaged) prediction vector is calculated by averaging the corresponding probabilities from each shard's prediction vector. The class with the highest average probability is chosen as the final prediction.

Strengths:

- Retains more information by considering the confidence scores across all classes; can improve accuracy compared to majority vote. Helps mitigate the impact of any single model's poor performance, as it averages out the confidence scores.

Weaknesses:

- More computationally intensive than simple majority voting because it involves handling and averaging multiple probability vectors. The individual models' probability outputs need to be well-calibrated to ensure that the averaged prediction vector provides meaningful confidence scores.

3. Training a Controller Model

Description: The Controller Model method is described as a technique where a secondary model, called the controller, is trained to re-weight the predictions made by constituent models. This controller model essentially learns which constituent model is best suited for predicting a given test point. The primary idea is to leverage the strengths of individual models by allowing the controller to determine the optimal combination of their outputs.

Strengths:

- The controller model can tailor the influence of each constituent model based on the specific characteristics of the input data, potentially leading to higher accuracy. By learning the optimal weights for different models, the controller can adapt to changes in the data distribution or the addition of new models.

Weaknesses:

- Introducing a controller model increases the complexity of the overall system, both in terms of computational requirements and implementation. As highlighted in the article, if the training data for the controller model needs to be unlearned, the entire controller must be retrained, which can be computationally expensive.

2. **Inefficiency Scenarios:** When will this method be inefficient to use? What do you suggest to mitigate this issue?

Answer:

The SISA algorithm can become inefficient under the following conditions:

1. Weak Learners in Small Shards:

When the training data is divided into too many small shards, each shard may not contain enough data to train a robust model. This can lead to weak learners that do not generalize well.

- **Mitigation:** Increase the size of each shard to ensure that the models have sufficient data to train effectively. Alternatively, use ensemble techniques to combine weak learners in a way that boosts their collective performance. Balance the number of shards and slices to minimize overhead while ensuring that each subset of data is manageable. Techniques such as cross-validation can be used to find an optimal partitioning strategy.

2. Retraining Costs for Unlearning:

When specific data points need to be unlearned, retraining the affected shards and slices can be computationally expensive. This is especially true if the data to be unlearned is spread across multiple shards and slices.

- **Mitigation:** Implement incremental learning techniques that allow for efficient updates to the model without full retraining. Alternatively, use a caching mechanism to store intermediate models' states to facilitate faster retraining.

3. Scalability Issues with Large Datasets:

As the dataset size increases, the number of shards and slices required also increases, leading to scalability issues. Managing and aggregating a large number of models can become impractical.

- **Mitigation:** Use parallel processing and distributed computing frameworks to handle large datasets. Ensure that the sharding and slicing processes are optimized for scalability.

4. High Overhead from Slicing and Aggregation:

The processes of slicing each shard and subsequently aggregating the models' outputs can introduce significant computational overhead, especially with large datasets or complex models.

- **Mitigation:** Optimize the number of slices and the aggregation process to balance performance and computational cost. Consider using more efficient data structures and algorithms to handle these steps.

3. **Performance Metrics:** What metric would you use to evaluate the performance of your unlearning algorithm? Reason your choice!

   Answer:

   To evaluate the performance of an unlearning algorithm, we would propose using the following metrics:

1. Retraining Time

Definition: Retraining time measures how quickly the model can be retrained after a request to unlearn specific data points.

Reason: This metric directly reflects the efficiency and practicality of the unlearning algorithm. Faster retraining times mean that the system can more quickly comply with unlearning requests, which is crucial for maintaining up-to-date models and adhering to user privacy requests.

2. Accuracy Degradation

Definition: Accuracy degradation measures the difference in model accuracy before and after the unlearning process.

Reason: Maintaining high accuracy is crucial for ensuring that the unlearning process does not negatively impact the overall performance of the model. This is particularly important for complex tasks where accuracy is a critical component.It is particularly useful in classification tasks to evaluate the performance of the unlearning algorithm. Since the goal of unlearning is to remove specific data points and adjust the model accordingly without significant loss in performance, accuracy can help in comparing the performance before and after unlearning.

4. **Model Architecture Impact:** Discuss the effect of the models' architecture on learning and unlearning time, and performance in both learning and unlearning.

   Answer:

   The architecture of a machine learning model significantly impacts both the learning and unlearning processes. Here are the key factors to consider:

1. Learning Time

- **Complexity:** Models with more complex architectures, such as deep neural networks with many layers, generally require more time to train. The number of parameters and the depth of the network increase the computational load.
- **Parallelism:** Architectures that support parallel processing (e.g., Convolutional Neural Networks) can reduce learning time by leveraging modern hardware like GPUs.

- **Optimization:** The choice of optimization algorithms and their compatibility with the model architecture can also affect learning time. For example, architectures that are compatible with gradient descent methods can benefit from faster convergence.

2. Unlearning Time

- **Modularity:** Architectures that allow for modular updates (e.g., models with isolated components or layers) can facilitate faster unlearning by only modifying the affected parts rather than the entire model.
- **Parameter Sharing:** Models that share parameters across different components may require more extensive updates during unlearning, potentially increasing the time required.
- **Simplicity:** Simpler architectures with fewer parameters generally require less time for unlearning, as fewer modifications are needed to remove the influence of specific data points.
- **Slicing:** (This method might not be specifically about architecture of the model, but we have mentioned it because of its importance in the article) By further dividing data dedicated for each model (i.e., each shard) and incrementally tuning (and storing) the parameter state of a model, we obtain additional time savings. It is also worth noting that the duration of training for the constituent models with and without data slicing is different when they have the same number of epochs. Each epoch takes less time when only a subset of the slices is being trained on; on the other hand, training incremental combinations of slices takes longer because the training process is repeated after each slice is added.

3. Learning Performance

- **Expressiveness:** More complex architectures, such as deep neural networks, tend to have higher expressiveness and can capture intricate patterns in data, leading to better performance.
- **Overfitting:** Complex models are also more prone to overfitting, especially if the training data is limited. Techniques like regularization are necessary to mitigate this issue.

4. Unlearning Performance

- **Resilience to Changes:** Architectures that are resilient to changes in their parameters can maintain performance after unlearning. For instance, models with robust regularization techniques are less likely to degrade in performance.
- **Localized Adjustments:** Models that allow for localized adjustments (e.g., through modular design) can effectively unlearn specific data points without significant impact on overall performance.
- **Impact on Structure:** The extent to which unlearning affects the model's structure and parameters can influence performance. Minimal and targeted changes tend to preserve performance better.

Private Training Models

Private training models are essential for protecting sensitive data during the machine learning process, especially against membership inference attacks. These models employ privacy-preserving techniques to prevent unauthorized access to individual data records. This is crucial for handling sensitive information like medical records or personal financial data.

Techniques for Private Training Models

- **Differentially Private Training:**
  - Adds noise to the training process to obscure the impact of individual data points.
  - Ensures that the removal or addition of a single data record does not significantly alter the model's output.
- **Regularization:**
  - Uses methods such as L1 or L2 norms to prevent overfitting by penalizing the magnitude of the model parameters.
  - Makes it harder for attackers to determine if a specific data point was used in the training set.
- **Normalization Temperature:**
  - Adjusts the normalization temperature to smooth the probability distribution over classes.
  - Reduces the model's sensitivity to individual data points, enhancing privacy.
- **Adding Noise for Privacy:**
  - Introduces randomness by adding noise to the dataset before training.
  - Protects individual data records from being identified.
  - Balances the amount of noise to maintain the model's performance while providing sufficient privacy protection.

These techniques help improve the robustness of machine learning models against information leakage, maintaining privacy without compromising utility.

Questions

5. **Differential Privacy:** Explain differentially private algorithms and their techniques for training a differentially private model.

Answer:

Differential privacy is a robust privacy standard that ensures the inclusion or exclusion of a single data point does not significantly affect the outcome of an analysis, thus providing privacy guarantees for individuals in the dataset.

Techniques for Training a Differentially Private Model

- **Noise Addition:**
  - **Laplace Mechanism:** Adds Laplace-distributed noise to the model's parameters or outputs. The amount of noise is calibrated to the sensitivity of the function being computed, ensuring that small changes in the input result in probabilistically bounded changes in the output.
  - **Gaussian Mechanism:** Adds Gaussian-distributed noise to the model's parameters or outputs. This mechanism is often used when the data or the function has a higher degree of smoothness and requires stronger privacy guarantees.
- **Differentially Private Stochastic Gradient Descent (DP-SGD):**
  - Modifies the standard Stochastic Gradient Descent (SGD) algorithm by adding noise to the gradients during training. Each gradient update is clipped to a predefined norm to limit the influence of any single data point, followed by the addition of noise to ensure differential privacy.

- **Private Aggregation of Teacher Ensembles (PATE):**
  - Uses an ensemble of teacher models, each trained on a disjoint subset of the data. The student model is trained on aggregated outputs of the teacher models, with noise added to the aggregation to ensure privacy. This method effectively balances privacy and utility by leveraging multiple models and adding noise at the aggregation step.
- **Objective Perturbation:**
  - Involves adding a random noise term to the objective function during model training. This ensures that the optimization process itself is privacy-preserving, as the added noise affects the training dynamics in a controlled manner.

Differentially private algorithms provide strong privacy guarantees by incorporating noise into the training process or the model outputs. Techniques such as the Laplace and Gaussian mechanisms, DP-SGD, PATE, and objective perturbation are crucial for training models that respect individual privacy while maintaining high utility.

6. **Regularization and Normalization:** Explain the regularization and normalization techniques used in training a private model. Are these techniques similar to the method of adding noise to the model in differential privacy?

Regularization techniques are used to prevent overfitting by penalizing model complexity. They help in making the model generalize better on unseen data.

- **L1 Regularization (Lasso):**
  - Adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function.
  - Can result in sparse models with fewer non-zero coefficients, promoting feature selection.
- **L2 Regularization (Ridge):**
  - Adds a penalty equal to the square of the magnitude of coefficients to the loss function.
  - Helps in preventing overfitting by keeping the model parameters small, promoting simpler models.
- **Elastic Net(Not mentioned in the articles):**
  - Combines both L1 and L2 regularization penalties.
  - Provides a balance between sparsity and the distribution of weights across features.

Normalization Techniques

Normalization techniques are used to scale the input data to a standard range or distribution, which can improve the training process and model performance.

- **Batch Normalization:**
  - Normalizes the input of each layer to have a mean of zero and a variance of one within a mini-batch.
  - Helps in accelerating the training process and stabilizing the learning by reducing internal covariate shift.
- **Layer Normalization:**
  - Normalizes the input across the features for each training example.
  - Useful in recurrent neural networks where batch normalization is less effective.

- **Instance Normalization:**
  - Normalizes each instance separately, often used in style transfer applications.
- **Group Normalization:**
  - Divides the channels into groups and normalizes each group independently.
  - Effective in small batch size settings.

Comparison with Differential Privacy Noise Addition

- **Similarity:**
  - Both regularization and noise addition in differential privacy involve modifying the training process to improve certain aspects of the model.
  - Regularization can be seen as implicitly adding noise to the model parameters by penalizing large weights.
- **Difference:**
  - The primary goal of regularization is to prevent overfitting and improve generalization, whereas the goal of noise addition in differential privacy is to protect individual data privacy.
  - Differential privacy explicitly adds random noise to the data or model outputs to achieve privacy guarantees, whereas regularization modifies the loss function to control model complexity.

While regularization and normalization techniques are primarily focused on improving model performance and generalization, they share some conceptual similarities with the noise addition used in differential privacy. Both approaches introduce modifications to the training process, but they serve different primary objectives: enhancing model robustness versus ensuring data privacy.

Answer:

7. **Additional Techniques:** Find other techniques for training a private model.

Answer:

In addition to the previously discussed methods such as Differentially Private Training, Regularization, and Adding Noise for Privacy, several other techniques can be employed to train private models. These techniques enhance data privacy and security in various ways.

- **Federated Learning:**
  - **Description:** In federated learning, multiple devices or servers collaboratively train a model without sharing their local data. Instead, they share model updates (gradients) with a central server that aggregates them to update the global model.
  - **Privacy Benefit:** Data remains decentralized, reducing the risk of data breaches and ensuring that raw data is never exposed.
- **Homomorphic Encryption:**
  - **Description:** Homomorphic encryption allows computations to be performed on encrypted data without decrypting it first. The results, when decrypted, match the output of operations performed on the plain data.
  - **Privacy Benefit:** Enables secure computation on sensitive data, ensuring that the data remains private even during processing.
- **Secure Multi-Party Computation (SMPC):**

  - **Description:** SMPC involves multiple parties jointly computing a function over their inputs while keeping those inputs private. Each party only knows its input and the final output.
  - **Privacy Benefit:** Ensures that no single party has access to all the data, maintaining privacy throughout the computation process.
- **Differential Privacy with Federated Learning:**
  - **Description:** Combines federated learning with differential privacy techniques by adding noise to the model updates before they are shared with the central server.
  - **Privacy Benefit:** Enhances the privacy guarantees of federated learning by ensuring that individual updates do not reveal sensitive information.
- **Adversarial Training:**
  - **Description:** Involves training the model with adversarial examples (intentionally perturbed inputs) to make the model more robust to such attacks.
  - **Privacy Benefit:** While primarily aimed at improving robustness, adversarial training can also enhance privacy by making it harder for adversaries to infer sensitive information from the model.

These additional techniques provide various methods for training private models, each with its own strengths and applications. By employing these methods, organizations can enhance the privacy and security of their machine learning models, protecting sensitive data from unauthorized access and inference attacks.

Membership Inference Attack

Introduction

Membership inference attacks are a type of privacy attack aimed at determining whether a particular data record was part of the training dataset of a machine learning model. These attacks pose significant risks, especially when the training data contains sensitive information.

Types of Membership Inference Attacks

Membership inference attacks are categorized based on the level of access the attacker has to the target model:

- **White-Box Attack:**
  - The attacker has complete knowledge of the target model, including its architecture, parameters, and training algorithm.
  - This allows for more precise attacks but requires high-level access.
- **Gray-Box Attack:**
  - The attacker has partial knowledge of the target model, such as some details about the architecture or training data.
  - Full access to the model's parameters is not available.
- **Black-Box Attack:**
  - The attacker has no knowledge of the target model's internals and interacts with it only through a public API.
  - This is the most common scenario and the focus of many research studies on membership inference attacks.

Paper's Method

The paper([2]) related to this project addresses the membership inference problem in a challenging black-box setting. The method involves:

- **Quantifying Membership Information Leakage:**
  - The attack model differentiates the target model's behavior on training inputs from its behavior on unseen inputs.
  - This transforms the membership inference problem into a classification problem.
- **Shadow Training Method:**
  - Several shadow models are generated to mimic the target model's behavior.
  - These models are trained on datasets similar to the target model's training dataset.
  - The attack model is then trained using the labeled inputs and outputs of these shadow models.

Questions

8. **Generating Shadow Training Data:** Explain three ways of generating training data for shadow models.

Answer: Shadow models are used to simulate the behavior of the target model, and generating appropriate training data for these models is crucial. Here are some ways to generate training data for shadow models:

1. Model-Based Synthesis:

This method involves generating synthetic training data for the shadow models using the target model itself. The process operates in two phases: search and sample. Search Phase: Uses a hill-climbing algorithm to find inputs that the target model classifies with high confidence. Sample Phase: Generates synthetic data from these high-confidence records until the training dataset for shadow models is full.

2. Statistics-Based Synthesis:

If the attacker has statistical information about the population from which the target model's training data was drawn, they can use this information to generate synthetic training records. Each feature value is sampled independently from its own marginal distribution.

3. Noisy Real Data

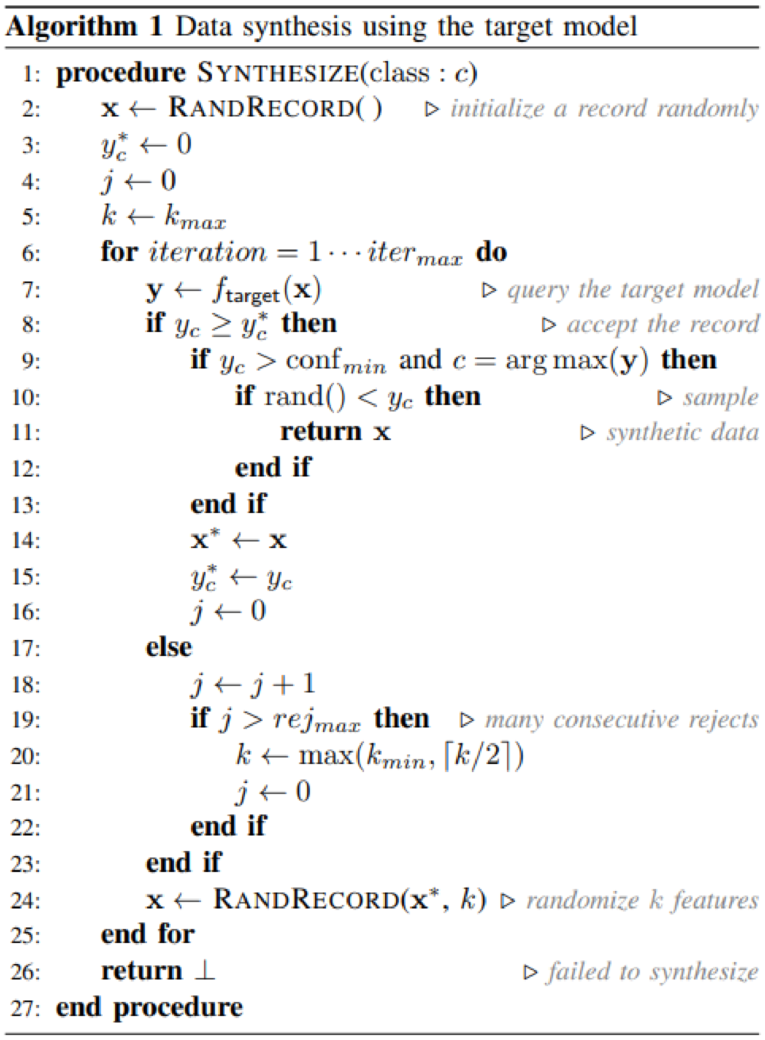
This method involves taking existing data similar to the target model's training data and adding noise to it. This approach assumes the attacker has access to data similar to the target model's training data, but it's "noisy" or not exact.For example, in experiments with location datasets, noisy data is simulated by flipping the binary values of 10% or 20% of the features and then training shadow models on this noisy dataset.

4. Using Public Datasets

Public datasets that have a similar distribution to the target model's training data are used to train shadow models.

9. **Synthetic Data Generation:** One of the method for generating training data for shadow models is using the model to generate synthetic data. Explain the Algorithm of synthetic data generation.

Answer:



Algorithm 1 Data synthesis using the target model

```
procedure SYNTHESIZE(class : $c$)
    $\mathbf{x} \leftarrow$ RANDRECORD( )    ▷ initialize a record randomly
    $y_c^* \leftarrow 0$
    $j \leftarrow 0$
    $k \leftarrow k_{max}$
    for $iteration = 1 \cdots iter_{max}$ do
        $\mathbf{y} \leftarrow f_{\text{target}}(\mathbf{x})$    ▷ query the target model
        if $y_c \geq y_c^*$ then    ▷ accept the record
            if $y_c > \text{conf}_{min}$ and $c = \arg\max(\mathbf{y})$ then
                if $\text{rand}() < y_c$ then    ▷ sample
                    return $\mathbf{x}$    ▷ synthetic data
                end if
            end if
            $\mathbf{x}^* \leftarrow \mathbf{x}$
            $y_c^* \leftarrow y_c$
            $j \leftarrow 0$
        else
            $j \leftarrow j + 1$
            if $j > rej_{max}$ then    ▷ many consecutive rejects
                $k \leftarrow \max(k_{min}, \lceil k/2 \rceil)$
                $j \leftarrow 0$
            end if
        end if
        $\mathbf{x} \leftarrow$ RANDRECORD($\mathbf{x}^*$, $k$)    ▷ randomize $k$ features
    end for
    return $\perp$    ▷ failed to synthesize
end procedure
```

Figure 1:

- **Initialization:**
  - A random record x is initialized.
  - y_c^* (best classification probability for class c) is set to 0.
  - k (number of features to change) is set to k_{max}.
  - j (rejection counter) is initialized to 0.

- **Main Loop:**
  - The loop runs for a maximum number of iterations ($iter_{\text{max}}$).

- In each iteration:
  - Query the target model to get the output $y = f_{\text{target}}(x)$.
  - If the target model's confidence y_c for class c is greater than the current best y_c^*:
    - If y_c exceeds the confidence threshold $conf_{\text{min}}$ and c is the predicted class, return x with a probability proportional to y_c.
    - Update x^* and y_c^* with the current record and confidence.
    - Reset the rejection counter j.
  - If the confidence is not better, increment j.
  - If j exceeds rej_{max}, reduce k and reset j.
- Randomize k features of x^* to generate a new record x.

- **Termination:**
  - If the loop completes without finding a suitable record, return $\perp$ indicating failure to synthesize the data.

10. **Training Attack Model:** Explain how attack model is trained using shadow models.

Answer:
Training the Attack Model:

(a) **Creating Shadow Models:**
- Shadow models are trained on datasets for which the attacker knows the ground truth. These datasets are similar to the target model's training data but are separate from it.

(b) **Labeling Data for Attack Model:**
- The attacker queries each shadow model with its training dataset and a disjoint test set of the same size.
- Outputs from the training dataset are labeled as "in".
- Outputs from the test set are labeled as "out".

(c) **Constructing the Attack Training Set:**
- For each record in the shadow model's training set, compute the prediction vector and add the record along with its output and "in" label to the attack training set.
- For each record in the shadow model's test set, compute the prediction vector and add the record along with its output and "out" label to the attack training set.

(d) **Partitioning and Training:**
- Split the attack training set into partitions, each associated with a different class label.
- Train a separate model for each class label that, given the prediction vector, predicts the membership status (in or out) for the input.

(e) **Model-Based Synthesis:**
- If using model-based synthesis, the training data for the attack model is derived from records classified by the target model with high confidence.
- This applies to both records in the shadow models' training sets and the test records left out of these datasets.
- The attack model learns to distinguish between high-confidence training inputs and high-confidence non-training inputs.

(f) **Binary Classification:**

- The task is converted into a binary classification problem: recognizing the relationship between training dataset members and the model's output.
- Any machine learning framework can be used for training the attack model. Examples include neural networks and black-box services like the Google Prediction API.

11. **Impact on Accuracy:** Explain the effect of following concepts in accuracy of the attack model

- Effect of the shadow training data generated using the three methods you explained earlier.
  Answer:
  Effect of the shadow training data generated using the three methods:
  The impact of shadow training data generation methods on the accuracy of an attack model is significant and varies based on the type of data used for training the shadow models. Here's a detailed discussion based on the provided text:

Effect of Noisy Shadow Training Data

Effective in scenarios where the noisy data closely resembles the target model's training data. Precision can be high, but accuracy may degrade if noise levels are too high.
When shadow models are trained on noisy versions of real data, as shown in Figure 2(figure 8 of the paper):

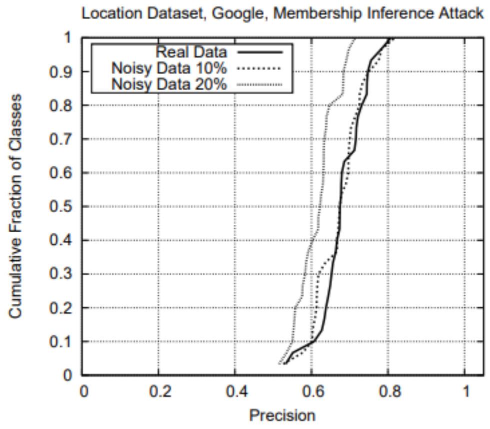


Fig. 8: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the location dataset. Results are shown for the shadow models trained on real data and for the shadow models trained on noisy data with 10% and 20% noise (i.e., $x\%$ of features are replaced with random values). Precision of the attack over all classes is 0.678 (real data), 0.666 (data with 10% noise), and 0.613 (data with 20% noise). The corresponding recall of the attack is 0.98, 0.99, and 1.00, respectively.

Figure 2:

- **Precision Drop with Increased Noise**: The precision of the attack decreases as the amount of noise in the shadow training data increases. However, even with a noise level of 10% (i.e., 10% of the features replaced with random values), the attack's precision matches the original attack trained on real data. This indicates that the attack model is robust to some degree of noise.
- **Robustness to Distribution Assumptions**: The attack model remains effective even if the assumptions about the distribution of the target model's training data are not entirely

accurate. This robustness is crucial for real-world scenarios where attackers may not have precise information about the target model's data distribution.

Effect of Synthetic Shadow Training Data

Figure 3(figure 9 of the paper) compares the precision of attacks when shadow models are trained on different types of synthetic data:

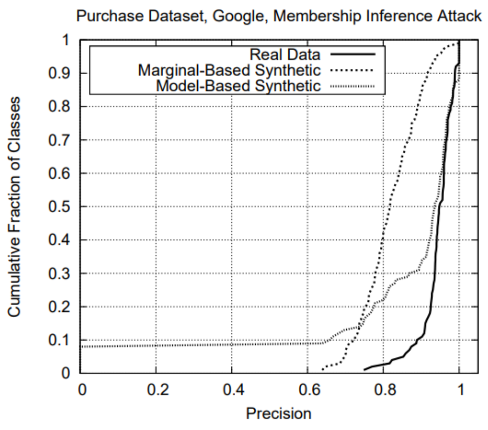


Fig. 9: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the purchase dataset. Results are shown for different ways of generating training data for the shadow models (real, synthetic generated from the target model, synthetic generated from marginal statistics). Precision of the attack over all classes is 0.935 (real data), 0.795 (marginal-based synthetic data), and 0.896 (model-based synthetic data). The corresponding recall of the attack is 0.994, 0.991, and 0.526, respectively.

Figure 3:

- **Marginal-Based Synthetic Data**: When shadow models are trained on synthetic data generated using marginal distributions of individual features:
  - **Precision**: The overall precision is 0.795, which is lower than the precision achieved with real data (0.935) but still significantly high.
  - **High Precision for Most Classes**: Despite the lower overall precision, the attack remains highly precise for most classes, indicating that marginal-based synthetic data can still be useful for training effective attack models.
- **Model-Based Synthetic Data**: Produces high precision in most classes but low precision in classes that are underrepresented in the target model's training dataset.

  When shadow models are trained on synthetic data generated using a model-based approach:
  - **Dual Behavior**: The precision of the attack exhibits a dual behavior. For most classes, the precision is high and close to the precision achieved with real data. However, for a few classes, the precision drops significantly (less than 0.1).
  - **Underrepresented Classes**: The low precision for some classes is attributed to the target classifier's inability to model the distribution of data records for these classes accurately. These underrepresented classes have fewer training examples in the target model's training dataset, making it challenging to synthesize accurate representatives.
- **Statistics-Based Synthesis Data**: Generally effective but less precise than model-based synthesis due to the lack of detailed knowledge about inter-feature correlations.

Key Insights

- **High Precision Achieved**: For the majority of the target model's classes, the attack achieves high precision, demonstrating that effective membership inference attacks can be trained with only black-box access to the target model.
- **No Prior Knowledge Required**: The attack can be successful without prior knowledge about the distribution of the target model's training data if the attacker can generate inputs classified with high confidence by the target model.

- Effect of the number of classes and training data per class.
  Answer:
  More classes can make it harder for the attack model to maintain high accuracy, especially if some classes are underrepresented. Adequate training data per class is crucial. Insufficient data for any class can lead to poor performance of the attack model for that class due to the difficulty in accurately modeling those classes.

  - **Number of Output Classes**:

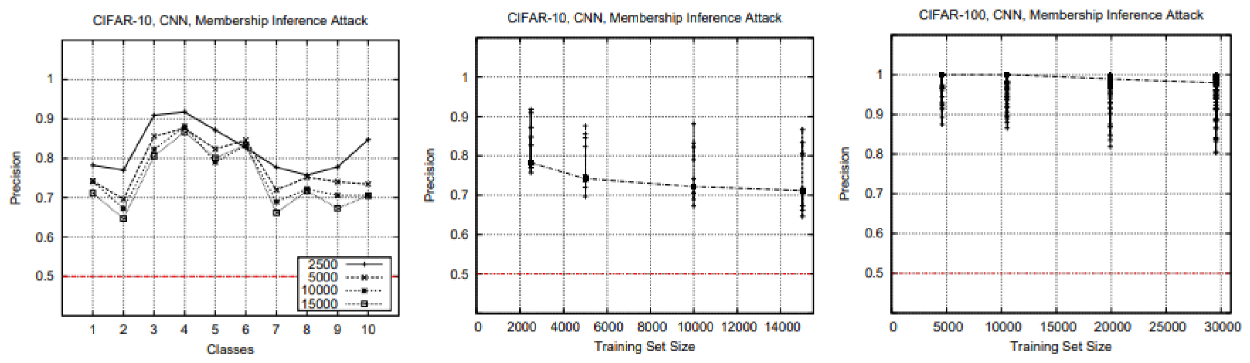


Fig. 4: Precision of the membership inference attack against neural networks trained on CIFAR datasets. The graphs show precision for different classes while varying the size of the training datasets. The median values are connected across different training set sizes. The median precision (from the smallest dataset size to largest) is 0.78, 0.74, 0.72, 0.71 for CIFAR-10 and 1, 1, 0.98, 0.97 for CIFAR-100. Recall is almost 1 for both datasets. The figure on the left shows the per-class precision (for CIFAR-10). Random guessing accuracy is 0.5.

Figure 4:

    * The number of output classes of the target model contributes to how much the model leaks. The more classes there are, the more signals about the internal state of the model are available to the attacker.
    * This explains why the results in Figure 4 are better for CIFAR-100 than for CIFAR-10. The CIFAR-100 model is also more overfitted to its training dataset.
    * For the same number of training records per class, the attack performs better against CIFAR-100 than against CIFAR-10.
    * **Example**: Comparing CIFAR-10 with a training dataset size of 2,000 and CIFAR-100 with a training dataset size of 20,000, the average number of data records per class is 200 in both cases. However, the attack accuracy is much better (close to 1) for CIFAR-100.
  - **Quantifying the Effect of Number of Classes**:
    * To understand this effect, target models were trained using Google Prediction API on the purchase dataset with {2, 10, 20, 50, 100} classes.
    * Figure 5(figure 10 of the paper) shows the distribution of attack precision for each model. Models with fewer classes leak less information about their training inputs.

* As the number of classes increases, the model needs to extract more distinctive features from the data to classify inputs with high accuracy. Models with more output classes need to remember more about their training data, thus they leak more information.

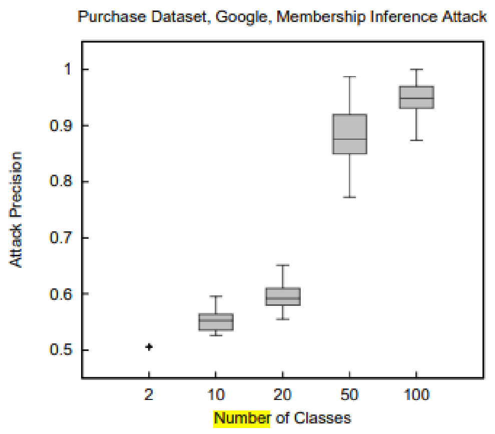


Fig. 10: Precision of the membership inference attack against different purchase classification models trained on the Google platform. The boxplots show the distribution of precision over different classification tasks (with a different number of classes).

Figure 5:

- **Amount of Training Data per Class**:

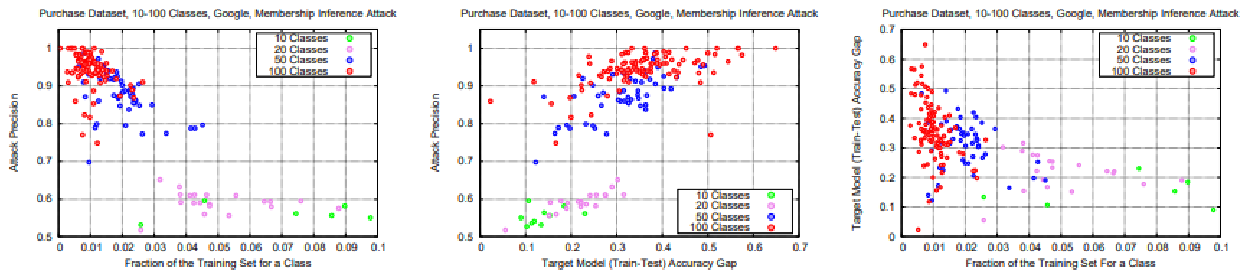
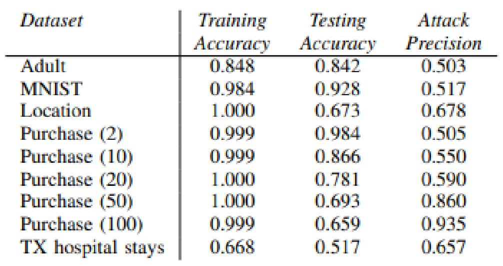


Fig. 11: Relationship between the precision of the membership inference attack on a class and the (train-test) accuracy gap of the target model, as well as the fraction of the training dataset that belongs to this class. Each point represent the values for one class. The (train-test) accuracy gap is a metric for generalization error [18] and an indicator of how overfitted the target model is.

Figure 6:

  * Figure 6(figure 11 of the paper) shows the relationship between the amount of training data per class and the accuracy of membership inference. This relationship is more complex, but generally, the more data associated with a given class in the training dataset, the lower the attack precision for that class.

- **Precision of Membership Inference**:
  * Table II shows the precision of membership inference against Google-trained models.
  * **For the MNIST dataset**:
    · Training accuracy of the target model: 0.984
    · Test accuracy of the target model: 0.928
    · Overall precision of the membership inference attack: 0.517 (slightly above random guessing)



Dataset	*Training Accuracy*	*Testing Accuracy*	*Attack Precision*
Adult	0.848	0.842	0.503
MNIST	0.984	0.928	0.517
Location	1.000	0.673	0.678
Purchase (2)	0.999	0.984	0.505
Purchase (10)	0.999	0.866	0.550
Purchase (20)	1.000	0.781	0.590
Purchase (50)	1.000	0.693	0.860
Purchase (100)	0.999	0.659	0.935
TX hospital stays	0.668	0.517	0.657

TABLE II: Accuracy of the Google-trained models and the corresponding attack precision.

        · The lack of randomness in the training data for each class and the small number of classes contribute to the failure of the attack.

    ∗ **For the Adult dataset**:
        · Training accuracy of the target model: 0.848
        · Test accuracy of the target model: 0.842
        · Overall precision of the attack: 0.503 (equivalent to random guessing)
        · Reasons for failure: The model is not overfitted (test and train accuracies are almost the same).
        · The model is a binary classifier, making it difficult for the attacker to distinguish members from non-members by observing the behavior of the model on essentially one signal, as the two outputs are complements of each other. This is not sufficient for the attack to extract useful membership information from the model.

- Effect of overfitting.

  Answer:

  Overfitting can significantly degrade the attack model's performance as it might capture noise and peculiarities of the shadow models' training data rather than generalizable patterns. To mitigate overfitting, techniques such as regularization, cross-validation, and ensuring a diverse and sufficiently large training dataset for shadow models are employed.

  – **Model Overfitting and Data Leakage**:
    ∗ The more overfitted a model is, the more it leaks information, but this is only true for models of the same type.
    ∗ For example, the Amazon-trained (100, $1e^{-4}$) model is more overfitted and leaks more information compared to the Amazon-trained (10, $1e^{-6}$) model.
    ∗ However, both Amazon models leak less information than the Google-trained model, despite the Google model being less overfitted than one of the Amazon models and having significantly better predictive power and generalizability.

  – **Overfitting vs. Model Structure**:
    ∗ Overfitting is not the sole factor that causes a model to be vulnerable to membership inference attacks.
    ∗ The structure and type of the model also contribute significantly to the vulnerability.

  – **Factors Contributing to Attack Accuracy**:
    ∗ In Figure 6, several factors contributing to attack accuracy per class are examined, including:
        · How overfitted the model is.
        · The fraction of the training data that belongs to each class.

    * The (train-test) accuracy gap is used as a metric to measure overfitting, which is the difference between the accuracy of the target model on its training and test data. Similar metrics are commonly used in the literature to measure overfitting.

- **Relationship Between Overfitting and Attack Precision**:
    * This metric is computed for each class, and larger gaps indicate that the model is more overfitted on its training data for that class.
    * The plots show that, as expected, larger (train-test) accuracy gaps are associated with higher precision of membership inference attacks.

References

1. Machine Unlearning - Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, Nicolas Papernot - 2019.
2. Membership Inference Attacks against Machine Learning Models - Reza Shokri, Marco Stronati, Congzheng Song, Vitaly Shmatikov - 2016.
3. Training Private Models That Know What They Don't Know - Stephan Rabanser1,2∗, Anvith Thudi1,2, Abhradeep Thakurta3, Krishnamurthy (Dj) Dvijotham3, Nicolas Papernot1,2 1University of Toronto, 2Vector Institute, 3Google DeepMind.