Phase 0

Mahdi Tabatabaei - 400101515
Heliya Shakeri - 400101391

Machine Learning

Dr. Amiri

June 25, 2024

Contents

Introduction

As we navigate through the vast ocean of data in this digital age, the power of machine learning and deep learning models has become increasingly evident. These models have revolutionized data analysis, providing us with insights that were previously unimaginable. However, with great power comes great responsibility. As we delve deeper into the world of data, one question becomes increasingly important: How do we ensure the privacy of the data we analyze?

This project aims to address this critical issue. We will explore various aspects of privacy in machine learning, focusing on two key areas.

In the first section, we will introduce you to the concept of *Machine Unlearning*, a process designed to uphold the right to be forgotten. This right is a crucial aspect of data privacy, ensuring that individuals can request their data to be removed from a model or system.

In the final section, we will delve into the realm of training private models. We will also discuss a specific adversarial attack known as the *Membership Inference Attack*. This attack poses a significant threat to data privacy, and understanding it is key to developing robust, privacy-preserving machine learning models.

By the end of this project, you will have a solid understanding of these important concepts and be well-equipped to navigate the complex landscape of privacy in machine learning. Let's start this journey together!

Machine Unlearning

Machine unlearning refers to the ability of a machine learning model to effectively forget or remove the influence of specific data points it has learned from, without the need to retrain the entire model from scratch. This capability is increasingly important due to several key reasons:

- **Privacy and Data Protection:** Regulations like the General Data Protection Regulation (GDPR) in Europe mandate that individuals have the right to request the deletion of their personal data. Machine unlearning helps in complying with such regulations by ensuring that a model can forget specific user data upon request.

- **Error Correction:** Sometimes, data used for training might contain errors or become irrelevant over time. Machine unlearning allows for the correction of these errors without the need for complete retraining.

- **Data Management:** In dynamic environments where data changes frequently, the ability to unlearn outdated or less relevant information can help maintain model performance and relevance.

SISA (Sharded, Isolated, Sliced, and Aggregated) Algorithm

The SISA algorithm is a structured approach designed to facilitate efficient machine unlearning. Here's a breakdown of how SISA works:

- **Sharding:** The training data is divided into multiple smaller subsets, or shards. Each shard contains a portion of the overall data. This ensures that the model training can be segmented into manageable parts.

- **Isolation:** Each shard is used to train a separate model or a component of a model independently. This isolation helps in minimizing the dependency across different parts of the training data.

- **Slicing:** Each shard is further divided into slices. Training occurs in a sequential manner, slice by slice, within each shard. This stepwise learning approach allows for finer control over the data's influence on the model.

Specifically, each shard's data D_k is further uniformly partitioned into R disjoint slices such that:

$$\bigcap_{i\in[R]} D_{k,i} = \emptyset \quad \text{and} \quad \bigcup_{i\in[R]} D_{k,i} = D_k$$

We perform training for e epochs to obtain M_k as follows:

- At step 1, train the model using random initialization using only $D_{k,1}$, for e_1 epochs. Let us refer to the resulting model as $M_{k,1}$. Save the state of parameters associated with this model.

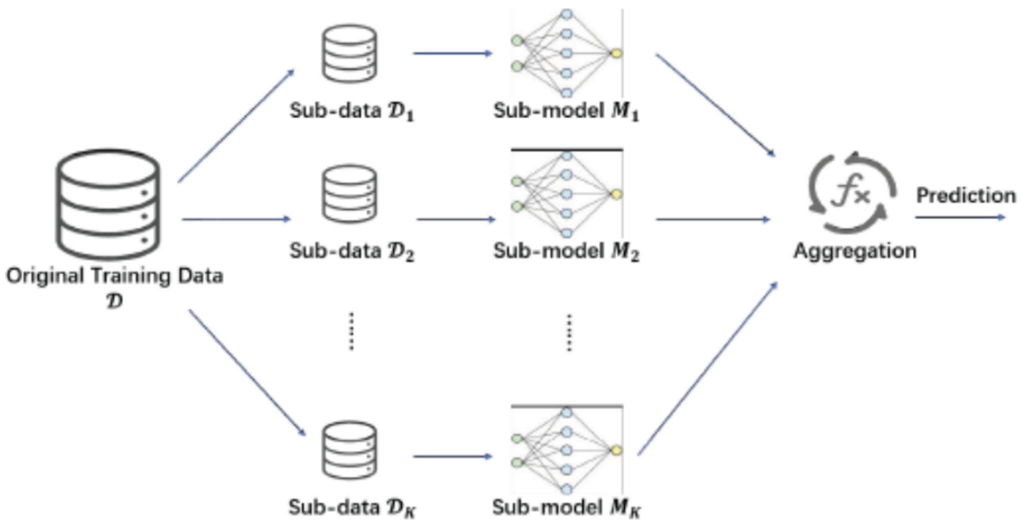
- At step 2, train the model $M_{k,1}$ using $D_{k,1} \cup D_{k,2}$, for e_2 epochs. Let us refer to the resulting model as $M_{k,2}$. Save the parameter state.

- At step R, train the model $M_{k,R-1}$ using $\cup_i D_{k,i}$ for e_R epochs. Let us refer to the resulting final model as $M_{k,R} = M_k$. Save the parameter state.

Aggregation: The final model is constructed by aggregating the outputs from the independently trained shards. This aggregation helps in forming a comprehensive model while maintaining the benefits of isolation and segmentation.

Unlearning with SISA

When unlearning is required, the SISA algorithm allows for the selective removal of data by:



- **Targeting Specific Shards:** Since the training data is sharded, only the shards containing the data to be forgotten need to be retrained or adjusted. This significantly reduces the computational cost compared to retraining the entire model.

- **Updating Relevant Slices:** Within the targeted shards, only the specific slices containing the data to be forgotten are modified. This granular approach ensures that changes are localized, preserving the integrity of the rest of the model.

- **Efficient Re-Aggregation:** After updating the necessary shards and slices, the outputs are re-aggregated to form the updated model. This allows the model to effectively "forget" the specified data points while retaining the knowledge from the remaining data.

The SISA algorithm thus provides an efficient and scalable method for machine unlearning, making it feasible to comply with privacy regulations and manage data integrity without the need for exhaustive retraining.

Questions

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

1. Majority Voting
   - **Strengths:**
     - Simple to implement and understand.
     - Reduces the risk of overfitting by considering multiple models.
   - **Weaknesses:**
     - May not work well if there are significant variations between the models.
     - Can be biased if some models are more accurate than others.
2. Weighted Averaging
   - **Strengths:**
     - Takes into account the performance of individual models by assigning higher weights to more accurate models.
     - Can provide a more balanced and accurate overall model.
   - **Weaknesses:**
     - Requires a reliable method to determine weights, which can be complex.
     - If weights are not appropriately assigned, it may lead to biased results.
3. Stacking (Meta-Learning)
   - **Strengths:**
     - Uses a meta-model to learn the best way to combine the outputs of base models, potentially leading to better performance.
   - **Weaknesses:**
     - Computationally intensive and requires more data and tuning.
     - The complexity of the meta-model can introduce new challenges and overfitting risks.

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

- Inefficiency Scenarios
  - **Large-Scale Data:** When dealing with extremely large datasets, the process of sharding, training, and aggregating can become computationally intensive and time-consuming.

- **High Model Complexity:** Complex models with many parameters can slow down the training and unlearning processes, making the method inefficient.
- **Frequent Unlearning Requests:** If there are frequent requests to unlearn data, the need to retrain specific shards and slices can lead to significant overhead.

- Mitigation Strategies
  - **Incremental Learning:** Implement incremental learning techniques to update the model with new data without fully retraining.
  - **Efficient Data Structures:** Use efficient data structures and algorithms to manage and update shards and slices.
  - **Parallel Processing:** Leverage parallel processing and distributed computing to handle large-scale data and complex models more efficiently.
  - **Batch Unlearning:** Group unlearning requests and process them in batches to reduce the frequency of retraining operations.

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

- Proposed Metric: Accuracy Retention and Forgetting Efficiency
  - **Accuracy Retention:** Measure the model's performance on a validation dataset before and after the unlearning process. The goal is to ensure that the model retains as much of its predictive accuracy as possible on the data it was not asked to forget.
  - **Forgetting Efficiency:** Evaluate the model's ability to effectively "forget" the data points it was asked to unlearn. This can be assessed by checking whether the model's predictions for the unlearned data points return to baseline levels (i.e., similar to the performance before the data points were included in training).
- Reason for Choice
  - **Comprehensive Evaluation:** Combining both accuracy retention and forgetting efficiency provides a comprehensive evaluation of the unlearning algorithm's performance, ensuring it balances both preserving useful knowledge and forgetting specific data points.
  - **Regulatory Compliance:** Helps in verifying compliance with data privacy regulations such as GDPR, which mandate the removal of personal data without compromising overall system performance.
  - **Model Integrity:** Ensures that the model remains robust and reliable even after the unlearning process, maintaining its integrity and usability in real-world applications.

4. Discuss the effect of the models' architecture on learning and unlearning time, and performance in both learning and unlearning.

- Learning Time
  - **Complexity:** More complex architectures with deeper layers and more parameters generally require longer training times due to the increased computational load.

  - **Efficiency:** Simplified architectures or those designed with efficient training mechanisms (e.g., convolutional neural networks for image data) can reduce learning time significantly.

- Unlearning Time
  - **Parameter Count:** Models with a higher number of parameters will take longer to adjust during the unlearning process as more weights need to be updated.
  - **Modularity:** Architectures that are modular or segmented (e.g., using techniques like SISA) can facilitate quicker unlearning by isolating the affected segments.

- Performance in Learning
  - **Expressiveness:** More complex architectures can capture more intricate patterns in the data, potentially leading to higher accuracy and better generalization.
  - **Overfitting Risk:** Increased complexity can also lead to overfitting, where the model performs well on training data but poorly on unseen data. Regularization techniques can mitigate this risk.

- Performance in Unlearning
  - **Adaptability:** Models that can quickly adapt to changes in the training data will perform better in unlearning scenarios. This includes architectures designed to accommodate incremental learning or dynamic updates.
  - **Residual Effects:** The architecture should ensure minimal residual effects of the unlearned data on the model's predictions, maintaining overall performance integrity.

Private Training Models

In the realm of machine learning, private training models are crucial for safeguarding sensitive data during the model training process. These models are designed with privacy-preserving techniques that prevent the exposure of individual data records to unauthorized parties. The necessity for private training models arises from the threat of membership inference attacks, where adversaries attempt to deduce whether a particular data record was used in training a machine learning model. By employing private training models, organizations can enhance the security of their machine learning systems, ensuring that the confidentiality of the training data is maintained and the privacy of individuals is protected against such invasive attacks. This is especially vital when dealing with sensitive information, such as medical records or personal financial data, where the implications of a privacy breach can be significant.

Here we explain some techniques for private training models:

- **Differentially Private Training:** This technique adds noise to the training process to obscure the contribution of individual data points, thus enhancing privacy. It ensures that the removal or addition of a single data record does not significantly affect the output of the model, providing a quantifiable level of privacy.

- **Regularization:** Regularization methods like L1 or L2 norms are used during training to prevent overfitting. By penalizing the magnitude of the model parameters, regularization makes it harder for attackers to infer whether a specific data point was used in the training set.

- **Normalization Temperature:** Increasing the normalization temperature involves adjusting the softmax function used in classification. A higher temperature leads to a smoother probability distribution over classes, which can reduce the model's sensitivity to individual data points and thus improve privacy.
- **Adding Noise for Privacy:** The concept of adding noise to a dataset involves intentionally introducing some level of randomness to the data before it is used to train a machine learning model. This technique is often employed as a privacy-preserving measure to protect individual data records from being identified or reconstructed. By altering the original data slightly, the risk of sensitive information being exposed through the model's predictions is reduced. However, it's crucial to balance the amount of noise added; too much can degrade the model's performance, while too little may not provide sufficient privacy protection. Effective noise addition can help maintain a model's utility while safeguarding the privacy of the individuals represented in the training dataset.

These techniques, among others, contribute to the robustness of machine learning models against membership inference attacks by reducing the risk of information leakage about the training data. Each method offers a different approach to enhancing privacy while maintaining the utility of the model.

▬ *Questions*

.

5. Explain differentially private algorithms and their techniques for training a differentially private model.

Differentially private algorithms are designed to provide privacy guarantees when analyzing and sharing data. The goal is to ensure that the output of an algorithm does not reveal specific information about any individual in the dataset, even if an adversary has access to all other data. This is achieved by adding controlled randomness to the algorithm, which obscures the contribution of any single data point.

- Key Concepts of Differential Privacy
  - **Differential Privacy (DP):** A mechanism is said to be differentially private if the inclusion or exclusion of a single data point does not significantly affect the output. Formally, a randomized algorithm $\mathcal{M}$ is ϵ-differentially private if for all datasets D_1 and D_2 differing on a single element, and for all subsets of outputs S:

    $$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D_2) \in S]$$

    where ϵ is a small positive parameter controlling the privacy loss.
  - **Privacy Budget (ϵ):** This parameter controls the trade-off between privacy and accuracy. Smaller values of ϵ provide stronger privacy but may lead to less accurate results.
- Techniques for Achieving Differential Privacy
  - **Noise Addition:** The most common technique is to add random noise to the output of a function. The noise is typically drawn from a Laplace or Gaussian distribution, scaled according to the sensitivity of the function (how much the output can change with the modification of a single data point).

    * **Laplace Mechanism:** Adds noise drawn from a Laplace distribution with mean 0 and scale $\Delta f/\epsilon$, where Δf is the sensitivity of the function f.
    * **Gaussian Mechanism:** Adds noise drawn from a Gaussian distribution with mean 0 and standard deviation proportional to $\Delta f/\epsilon$.
  - **Sensitivity Analysis:** This involves calculating the sensitivity of a function, which measures how much the function's output can change when a single data point is added or removed. Functions with lower sensitivity require less noise to achieve differential privacy.
  - **Sampling:** Randomly sampling subsets of the data can reduce the impact of any single data point, contributing to differential privacy. Techniques like subsampling can be combined with noise addition to enhance privacy.

- Techniques for Training Differentially Private Models
  - **Differentially Private Stochastic Gradient Descent (DP-SGD):** A variant of the standard stochastic gradient descent algorithm used in training machine learning models. It involves adding noise to the gradients computed during each iteration to ensure privacy. The process involves:
    * Clipping the gradients to a maximum norm to limit sensitivity.
    * Adding noise to the clipped gradients before updating the model parameters.
    * Aggregating the noisy gradients over multiple iterations to ensure the overall process is differentially private.
  - **Private Aggregation of Teacher Ensembles (PATE):** This technique trains multiple models (teachers) on disjoint subsets of the data. The predictions of these models are aggregated in a differentially private manner to train a student model. The student model learns from the noisy aggregated predictions, which provide privacy guarantees.
  - **Private Variational Autoencoders (PVAEs):** This approach involves training variational autoencoders with differential privacy by adding noise to the encoder's output or the decoder's parameters. The privacy is ensured through the noise addition and careful tuning of the privacy budget.
  - **Private Federated Learning:** Combines federated learning (where multiple participants train models on their local data) with differential privacy. Noise is added to the model updates from each participant before they are aggregated, ensuring that the global model is differentially private.

- Practical Considerations
  - **Choosing the Privacy Budget (ϵ):** A smaller ϵ offers better privacy but can degrade model performance. The choice of ϵ depends on the acceptable trade-off between privacy and accuracy.
  - **Calibration of Noise:** The amount and distribution of noise should be carefully calibrated based on the sensitivity of the function and the desired level of privacy.
  - **Utility vs. Privacy Trade-off:** It is important to balance the privacy guarantees with the utility of the model. Excessive noise can render the model useless, while too little noise might not provide sufficient privacy.

  - **Implementation and Tools:** Libraries like TensorFlow Privacy and PySyft provide tools for implementing differentially private algorithms, making it easier to integrate these techniques into machine learning workflows.

By incorporating these techniques, differentially private algorithms enable the training of models that provide strong privacy guarantees while maintaining useful performance.

6. 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 and normalization are common techniques used in training machine learning models to improve their performance and generalization. These techniques help prevent overfitting and ensure that the model learns robust features. While these techniques share some conceptual similarities with noise addition in differential privacy, their purposes and implementations are different.

Regularization Techniques:

Regularization techniques are used to add constraints or penalties to the training of a machine learning model to prevent overfitting. Overfitting occurs when a model learns not only the underlying patterns but also the noise in the training data, resulting in poor generalization to new data.

- L1 Regularization (Lasso)
  - Adds a penalty equal to the absolute value of the magnitude of coefficients.
  - Encourages sparsity, meaning it drives some coefficients to be exactly zero, effectively performing feature selection.
  - Objective function:

$$\mathcal{L}(\theta) + \lambda \sum_i |\theta_i|$$

- L2 Regularization (Ridge)
  - Adds a penalty equal to the square of the magnitude of coefficients.
  - Encourages smaller coefficients, thus spreading out the weights more evenly and reducing the risk of overfitting.
  - Objective function:

$$\mathcal{L}(\theta) + \lambda \sum_i \theta_i^2$$

- Elastic Net Regularization
  - Combines L1 and L2 regularization.
  - Balances between the sparsity of L1 and the smoothness of L2.
  - Objective function:

$$\mathcal{L}(\theta) + \lambda_1 \sum_i |\theta_i| + \lambda_2 \sum_i \theta_i^2$$

- Dropout
  - Randomly drops units (along with their connections) from the neural network during training.

  - Prevents units from co-adapting too much and forces the network to learn more robust features.
  - Typically, a dropout rate (e.g., 0.5) specifies the probability of dropping a unit.

Normalization Techniques:

Normalization techniques aim to standardize the input data to have specific properties, such as a mean of zero and a variance of one. This helps improve the convergence speed of gradient descent algorithms and ensures that all features contribute equally to the learning process.

- Min-Max Scaling
  - Scales the data to a fixed range, usually [0, 1].
  - Formula:
$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$
- Standardization (Z-score Normalization)
  - Centers the data to have a mean of zero and a standard deviation of one.
  - Formula:
$$X' = \frac{X - \mu}{\sigma}$$
- Batch Normalization
  - Normalizes the inputs of each layer so that they have a mean of zero and a variance of one during training.
  - Helps stabilize and accelerate the training process in deep neural networks.
  - Formula (for a batch B of inputs X):
$$\hat{X} = \frac{X - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$
    where μ_B and σ_B^2 are the mean and variance of the batch, and ϵ is a small constant.
- Layer Normalization
  - Normalizes the inputs across the features for each training example independently.
  - Used mainly in recurrent neural networks.

Comparison to Differential Privacy Noise Addition

While both regularization and noise addition in differential privacy involve modifying the model training process, their goals and methods differ significantly:

- Purpose
  - Regularization aims to improve model generalization and prevent overfitting.
  - Normalization aims to standardize input data to facilitate better training dynamics.

  - Noise addition in differential privacy aims to protect individual data points' privacy by adding controlled randomness to the model's output or parameters.

- Implementation
  - Regularization involves adding penalties to the loss function or altering the model architecture (e.g., dropout).
  - Normalization involves transforming the input data or intermediate layer outputs.
  - Noise addition in differential privacy involves adding noise to the gradients, outputs, or parameters, with the amount of noise calibrated based on the desired privacy level.

- Effect on Model
  - Regularization and normalization improve the model's robustness and training efficiency.
  - Noise addition in differential privacy may degrade the model's accuracy to some extent but provides formal privacy guarantees.

7. Find other techniques for training a private model.

There are several techniques beyond the commonly discussed ones for training private models, each with its own strengths and specific use cases. Here are some additional methods:

- Federated Learning

  Federated Learning is a distributed approach where multiple clients (e.g., mobile devices) collaboratively train a model while keeping the data localized. Each client trains the model on its local data and sends only the model updates (gradients) to a central server. The server aggregates these updates to improve the global model. This technique enhances privacy by ensuring that raw data never leaves the client's device.

  - **Advantages:** Protects raw data privacy, scales well with large numbers of clients, and reduces the risk of data breaches.
  - **Disadvantages:** Requires robust communication infrastructure, susceptible to model inversion attacks if not combined with differential privacy.

- Homomorphic Encryption Homomorphic Encryption allows computations to be performed on encrypted data without decrypting it. This ensures that the data remains confidential throughout the processing.

  - **Advantages:** Provides strong privacy guarantees since data remains encrypted during computation.
  - **Disadvantages:** Computationally expensive and slower compared to unencrypted computations, making it less practical for real-time applications.

- Secure Multi-Party Computation (SMPC)

  SMPC involves multiple parties jointly performing computations on their combined data while keeping each party's input private. Each party holds a share of the data, and the computation is carried out in such a way that no party learns anything about the others' data.

  - **Advantages:** Strong privacy guarantees as no single party can access the complete dataset.
  - **Disadvantages:** High computational and communication overhead, complexity in implementation.

- Local Differential Privacy (LDP)

  In LDP, the data is randomized before it is collected, ensuring that the data collector never sees the original data. Each user's data is perturbed locally, providing privacy at the individual level.

  - **Advantages:** Strong privacy guarantees since data is anonymized before transmission, easy to implement.
  - **Disadvantages:** The noise added locally can significantly reduce the accuracy of the aggregated results.

- Private Aggregation of Teacher Ensembles (PATE)

  PATE trains multiple teacher models on disjoint subsets of the data. A student model is then trained on the aggregated (and noisy) outputs of these teachers. This technique leverages the privacy guarantees of ensemble learning and the aggregation process.

  - **Advantages:** Effective for tasks where ensemble learning is beneficial, good privacy guarantees when properly implemented.
  - **Disadvantages:** Requires careful tuning of the noise addition process, can be computationally intensive.

- Adversarial Training for Privacy

  Adversarial training involves training a model to be resistant to privacy attacks. This is done by introducing adversarial examples (inputs designed to deceive the model) during training to make the model more robust to potential attacks.

  - **Advantages:** Enhances the model's robustness and can protect against certain types of privacy attacks.
  - **Disadvantages:** Complex to implement, requires a balance between robustness and accuracy.

- TEE-Based Secure Enclaves

  Trusted Execution Environments (TEEs) like Intel SGX allow sensitive computations to be performed in a secure enclave. The data and code inside the enclave are protected from external access, even by the system's own operating system.

  - **Advantages:** Provides a high level of security for sensitive computations, relatively easy to use with existing hardware support.
  - **Disadvantages:** Limited by hardware capabilities, potential vulnerabilities in TEE implementations.

Membership Inference Attack

Introduction

Membership inference attacks are a type of privacy attack against machine learning models. The goal of these attacks is to determine whether a particular data record was used in the training dataset of a machine learning model. This can be problematic, especially when the training data contains sensitive information.

Types of Membership Inference Attacks

There are three main types of membership inference attacks, categorized based on the level of access the attacker has to the target model:

White-Box Attack

In a white-box attack, the attacker has complete knowledge of the target model, including its architecture, parameters, and training algorithm. This allows for a more precise attack but requires a high level of access that may not always be available.

Gray-Box Attack

A gray-box attack assumes partial knowledge of the target model. The attacker may know some details about the model's architecture or training data but does not have full access to the model's parameters.

Black-Box Attack

In a black-box attack, the attacker has no knowledge of the target model's internals and can only interact with it through a public API. This is the most common scenario and the focus of many research studies on membership inference attacks.

Paper's Method

We address the membership inference problem which is determining if a record was part of a machine learning model's training dataset. This problem is investigated in a challenging setting where the adversary only has black-box access to the model. We quantify membership information leakage via the model's prediction outputs. To solve this, we train an attack model to differentiate the target model's behavior on training inputs from its behavior on unseen inputs, effectively transforming the membership inference problem into a classification problem.

Attacking black-box models such as those built by commercial "machine learning as a service" providers requires more sophistication than attacking white-box models whose structure and parameters are known to the adversary.

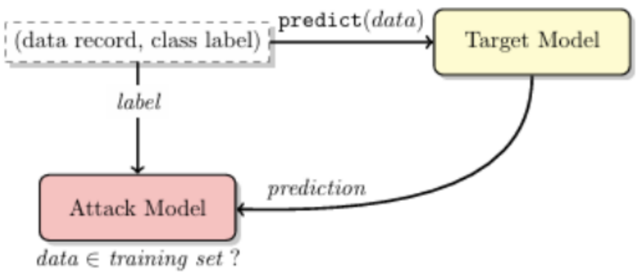


Figure 1: The attacker queries the target model with a data record and obtains the model's prediction on that record. The prediction is a vector of probabilities, one per class, that the record belongs to a certain class. This prediction vector, along with the label of the target record, is passed to the attack model, which infers whether the record was in or out of the target model's training dataset.

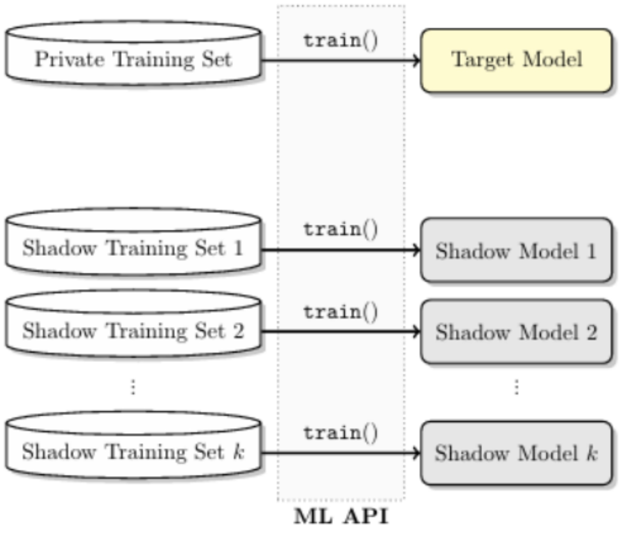


Figure 2: Training shadow models using the same machine learning platform as was used to train the target model. The training datasets of the target and shadow models have the same format but are disjoint. The training datasets of the shadow models may overlap. All models' internal parameters are trained independently.

To devise our attack models, we employ a novel shadow training method. Initially, we generate several "shadow models" that mimic the target model's behavior. For these models, we have knowledge of the training datasets and hence, the ground truth about dataset membership. Subsequently, the attack model is trained using the labeled inputs and outputs of these shadow models.

Questions

.

8. Explain three ways of generating training data for shadow models.

There are three primary methods for generating training data for shadow models in the context of membership inference attacks:

1. **Using Synthetic Data**:

- Generate synthetic data by sampling from the same distribution as the target model's training data.
- This involves creating data points that follow the same statistical properties and patterns as the original dataset.
- Synthetic data can be generated using techniques such as Gaussian distributions or other probabilistic models that approximate the target dataset's characteristics.

$$\text{Synthetic Data} \sim \mathcal{D}_{\text{target}}$$

2. **Using Noisy Data**:

- Create noisy versions of the original training data by adding perturbations or noise.
- This method involves altering the original data points by introducing controlled random noise, ensuring the modified data points still resemble the original ones closely.
- Noise can be added using various techniques like Gaussian noise or Laplacian noise.

$$\text{Noisy Data} = \text{Original Data} + \mathcal{N}(\mu, \sigma^2)$$

3. **Using Auxiliary Data**:

- Utilize an auxiliary dataset that is similar to the target model's training data.
- The auxiliary data should have similar features and distribution to the target dataset but can be sourced from publicly available data or other datasets with comparable characteristics.
- This approach leverages the assumption that the shadow models trained on auxiliary data can approximate the behavior of the target model trained on its original data.

$$\text{Auxiliary Data} \sim \mathcal{D}_{\text{aux}}$$

9. One of the methods for generating training data for shadow models is using the model to generate synthetic data. Explain the Algorithm of synthetic data generation.

Algorithm for Generating Synthetic Data Using the Model

To generate synthetic data for training shadow models, we can use the target model itself to create data points. The following algorithm outlines the steps for synthetic data generation:

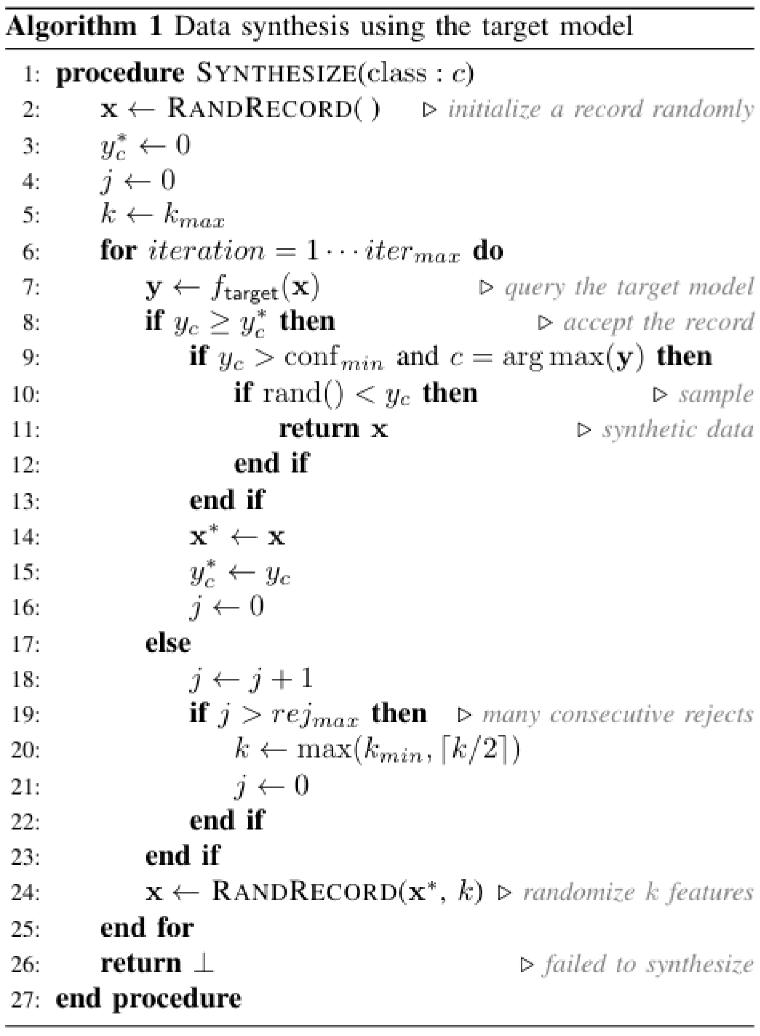
Algorithm 1 Synthetic Data Generation Using the Target Model

```
Require: Target model $M$, number of synthetic data points $n$
Ensure: Synthetic dataset $S_{\text{synthetic}}$
1: Initialize an empty dataset $S_{\text{synthetic}} \leftarrow \emptyset$
2: for $i = 1$ to $n$ do
3:     Sample a random input $x_i$ from the input space $\mathcal{X}$
4:     Obtain the output $y_i = M(x_i)$ from the target model
5:     Add the pair $(x_i, y_i)$ to the synthetic dataset $S_{\text{synthetic}}$
6: end for
7: return $S_{\text{synthetic}}$
```

Detailed Steps

1. **Initialization**: Start with an empty dataset $S_{\text{synthetic}}$.
2. **Sampling Inputs**: For each data point to be generated, sample a random input x_i from the input space $\mathcal{X}$. The input space $\mathcal{X}$ can be defined based on the feature space of the target model.
3. **Model Prediction**: Pass the sampled input x_i through the target model M to obtain the output y_i. This output y_i could be a class label in the case of classification tasks or a continuous value for regression tasks.
4. **Data Collection**: Add the input-output pair (x_i, y_i) to the synthetic dataset $S_{\text{synthetic}}$.
5. **Iteration**: Repeat the process until the desired number of synthetic data points n is generated.
6. **Return Synthetic Dataset**: Finally, return the generated synthetic dataset $S_{\text{synthetic}}$.

This algorithm leverages the target model to generate data points that follow the same distribution as the model's training data. By sampling random inputs and using the model's predictions, we can create a synthetic dataset that mimics the behavior of the original training data, enabling effective training of shadow models.



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 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 3

10. Explain how the attack model is trained using shadow models.

Training the Attack Model Using Shadow Models To train an attack model for membership inference, shadow models are employed to simulate the behavior of the target model. The process involves several steps:

- Training Shadow Models
  - **Data Partitioning**: Divide the available dataset into multiple subsets.
  - **Shadow Model Training**: Train multiple shadow models on these subsets. Each shadow model is trained on a different subset, allowing it to capture diverse aspects of the target model's behavior.
  - **Labeling Data**: For each shadow model, label the training data points as members and the non-training data points as non-members. This provides a ground truth for membership status.
- Generating Query Responses
  - **Querying Shadow Models**: Query each shadow model with both its training data and a separate set of non-training data.
  - **Collecting Outputs**: Collect the outputs (predictions and confidence scores) from the shadow models for both members and non-members.
- 3. Training the Attack Model
  - **Feature Extraction**: Extract features from the outputs of the shadow models. These features can include prediction confidence, probability distributions, and other model-specific metrics.
  - **Labeling Features**: Label the extracted features based on the membership status (member or non-member).
  - **Attack Model Training**: Use the labeled features to train a binary classifier, which serves as the attack model. This classifier learns to distinguish between the features of members and non-members.

Algorithm 2 Training the Attack Model Using Shadow Models

```
Require: Target model $M$, dataset $D$, number of shadow models $k$
Ensure: Trained attack model $A$
 1: Initialize an empty dataset $D_{\text{attack}} \leftarrow \emptyset$
 2: for $i = 1$ to $k$ do
 3:     Randomly partition $D$ into training subset $D_i^{\text{train}}$ and non-training subset $D_i^{\text{non-train}}$
 4:     Train shadow model $S_i$ on $D_i^{\text{train}}$
 5:     for each $x_j \in D_i^{\text{train}}$ do
 6:         Query $S_i(x_j)$ to obtain output $o_j$
 7:         Label $o_j$ as member
 8:         Add $(o_j, \text{member})$ to $D_{\text{attack}}$
 9:     end for
10:     for each $x_j \in D_i^{\text{non-train}}$ do
11:         Query $S_i(x_j)$ to obtain output $o_j$
12:         Label $o_j$ as non-member
13:         Add $(o_j, \text{non-member})$ to $D_{\text{attack}}$
14:     end for
15: end for
16: Train attack model $A$ on $D_{\text{attack}}$
17: return $A$
```

Detailed Steps

1. **Training Shadow Models**:
   - Split the available dataset into multiple subsets to train k shadow models.
   - For each subset, train a shadow model and label the data points as members or non-members based on whether they were in the training set of that shadow model.
2. **Generating Query Responses**:
   - Query each shadow model with both its training (member) data and a separate set of non-training (non-member) data.
   - Collect the output responses (e.g., prediction probabilities) for both member and non-member data points.
3. **Training the Attack Model**:
   - Extract features from the collected outputs, such as confidence scores or probability distributions.
   - Label the features based on whether they correspond to member or non-member data.
   - Train a binary classification model on the labeled features to create the attack model, which can now predict the membership status of data points based on the target model's outputs.

By following this process, the attack model can effectively learn to infer membership status, exploiting the information leaked through the outputs of the shadow models.

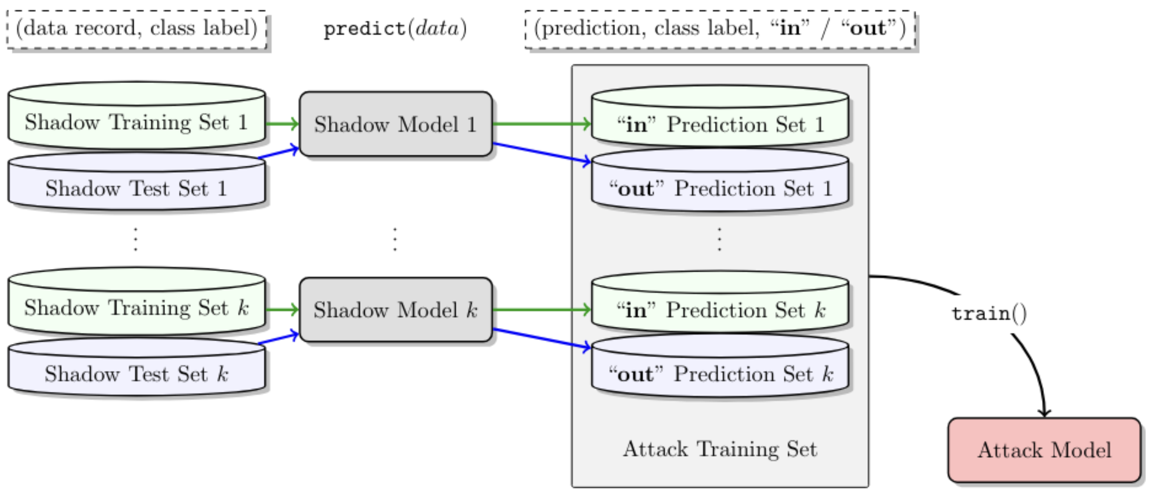


Figure 4

11. Explain the effect of the following concepts in accuracy of the attack model:
(a) Effect of the shadow training data generated using the three methods you explained earlier.
(b) Effect of the number of classes and training data per class.
(c) Effect of overfitting.

- Effect of the Shadow Training Data Generated Using the Three Methods
  - **Model-based synthesis:** Generates high-quality synthetic data that closely resembles the target model's training data, resulting in higher accuracy of the attack model. This method ensures that the synthesized data is highly representative of the target data, thus leading to more effective attack models. Achieves a precision of 0.895, balancing between high precision and some vulnerabilities in underrepresented classes.
  - **Statistics-based synthesis:** This method may not capture the joint distribution of the features accurately, leading to moderate accuracy. The generated data might miss some of the complex relationships between features present in the real training data, resulting in less effective attack models compared to model-based synthesis. Precision drops to 0.795 but remains effective, demonstrating robustness even with less accurate data generation methods.
  - **Noisy real data:** If the noise level is not too high, this method can produce reasonably accurate shadow models. The generated data retains much of the structure of the original data, making the attack model effective. However, too much noise can degrade the accuracy of the shadow models and, consequently, the attack model.

- Effect of the Number of Classes and Training Data per Class
  - **Number of Classes:**
    - **Process:** The target model's number of output classes impacts how much information it leaks. Models with more output classes need to extract more distinctive features to classify inputs accurately.
    - **Accuracy Impact:** The more classes the target model has, the more signals about its internal state are available to the attacker. This makes the model more

vulnerable to membership inference attacks. For example, CIFAR-100, with more classes, is more vulnerable than CIFAR-10 because the model needs to remember more about the training data, increasing the leakage of information.

  - **Training Data per Class:**
    - **Process:** The amount of training data associated with each class affects the attack's precision.
    - **Accuracy Impact:** Sparse data per class can lead to lower accuracy as the attack model struggles to learn the distribution effectively. Conversely, a larger amount of training data per class can lower the attack precision for that class because the model does not overfit as much. Overfitting to smaller classes provides more distinctive patterns for the attack model to exploit.

- Effect of Overfitting
  - **Process:** Overfitting occurs when a model performs significantly better on its training data than on unseen test data. This typically happens when the model memorizes the training data rather than learning generalizable patterns.
  - **Accuracy Impact:** Overfitting in the target model increases the success of membership inference attacks. Overfitted models tend to memorize the training data, making it easier for the attack model to distinguish between in and out samples. The model's confident predictions on training data versus less confident predictions on unseen data provide a clear signal for the attack model. However, overfitting is not the only factor contributing to vulnerability; the model's structure and type also play significant roles. Experiments show that attacks on overfitted models, such as CIFAR-10 and CIFAR-100, yield higher precision and recall. For instance, the test accuracy for overfitted neural-network models was 0.6 and 0.2 for CIFAR-10 and CIFAR-100, respectively, indicating high overfitting and thus high vulnerability to membership inference attacks.