Introduction to Machine Learning Project

Instructor: Prof. Rouhollah Amiri

Sharif University of Technology

All you need is Privacy!

Mohammad Mahdi Razmjoo 400101272

Parsa Hatami 400100962

Summer 2024

Simulation Question 1:

The theoretical basis of implementation is as follows:

• **SISA Training:**

SISA (Sharded, Isolated, Sliced, and Aggregated) training involves dividing the dataset into multiple shards (S). Each shard is then further divided into slices (R). The model is trained incrementally on each slice within a shard. This method enhances data privacy and enables efficient machine unlearning.

• **Machine Unlearning:**

Machine unlearning aims to remove the influence of specific data points from a trained model without retraining from scratch. The SISA method facilitates efficient unlearning by allowing the deletion of specific slices or shards and retraining only those parts, rather than the entire dataset.

• **Membership Inference Attacks:**

These attacks determine whether a specific data point was used in training the model. Evaluating the model's resilience to such attacks involves training with different levels of data granularity and observing performance metrics.

Output Analysis:

Majority Vote vs Averaging Vectors:

The averaging vectors method consistently performs better than the majority vote method in terms of F1-score, accuracy, precision, recall, and AUROC. This indicates that averaging the output vectors of models leads to more accurate and reliable predictions.

Impact of Sharding (S) and Slicing (R):

Performance Trend: As the number of shards (S) and slices (R) increases, performance metrics generally decrease. This is likely due to the reduced data size per shard and slice, which can lead to less effective training due to insufficient data in each training segment.

Optimal Configuration: Smaller values of S and R (e.g., S=5, R=5) tend to perform better, suggesting a balance between the granularity of data division and the availability of sufficient training data per segment.Results for Majority Vote:

S	R	F1-Score	Accuracy	Precision	Recall	AUROC
5	5	0.8293	0.8303	0.8347	0.8303	-
5	10	0.8227	0.8229	0.8264	0.8229	-
5	20	0.8099	0.8095	0.8138	0.8095	-
10	5	0.8160	0.8153	0.8186	0.8153	-
10	10	0.7713	0.7743	0.7758	0.7743	-
10	20	0.7527	0.7507	0.7585	0.7507	-
20	5	0.7727	0.7714	0.7832	0.7714	-
20	10	0.6803	0.6900	0.6954	0.6900	-
20	20	0.6722	0.6716	0.6835	0.6716	-

Results for Averaging Prediction Vectors:

S	R	F1-Score	Accuracy	Precision	Recall	AUROC
5	5	0.8490	0.8495	0.8537	0.8495	0.9871
5	10	0.8378	0.8378	0.8421	0.8378	0.9851
5	20	0.8231	0.8227	0.8280	0.8227	0.9821
10	5	0.8337	0.8328	0.8362	0.8328	0.9835
10	10	0.7853	0.7875	0.7906	0.7875	0.9761
10	20	0.7664	0.7634	0.7732	0.7634	0.9706
20	5	0.7820	0.7806	0.7909	0.7806	0.9754
20	10	0.6841	0.6924	0.7008	0.6924	0.9552
20	20	0.6879	0.6865	0.7004	0.6865	0.9528

S=5, R=5: Highest performance across all metrics, indicating the effectiveness of this configuration in maintaining model accuracy and robustness.
S=20, R=20: Lowest performance, highlighting the trade-off between privacy (more shards and slices) and model performance.

Simulation Question 2:

The theoretical basis of implementation is as follows:

• **SISA Unlearning Algorithm:**

The SISA (Sharding, Isolation, Slicing, Aggregation) algorithm is designed to facilitate efficient machine unlearning, where a machine learning model can effectively "forget" specific data points without retraining from scratch. This is crucial for compliance with data privacy regulations and for enhancing model performance by removing erroneous or outdated data.

Key Concepts:

1.Sharding: The dataset is divided into smaller subsets (shards). Each shard is used to train a separate model or part of a model, reducing dependencies across data subsets.

2.Isolation: Each shard is processed independently, minimizing the influence of data in one shard on another.

3.Slicing: Each shard is further divided into slices. Training occurs sequentially within each shard slice by slice, allowing fine-grained control over data influence.

4.Aggregation: The final model is constructed by aggregating outputs from independently trained shards, maintaining the benefits of isolation and segmentation.

• **Membership Inference Attacks (MIA):**

MIA aims to determine if a specific data point was part of the training dataset by analyzing the model's predictions. These attacks leverage the differences in model behavior between seen (training) and unseen (non-training) data, posing privacy risks.

Attack Model:

Shadow Models: Multiple models trained on datasets that mimic the target model's training data distribution.

Training Attack Model: Using the predictions from shadow models, an attack model is trained to classify data points as members or non-members of the training dataset.

Output Analysis:

Original Metrics (Majority Vote): f1 : 0.8316844922281084 accuracy : 0.8313 precision : 0.8349764958892995 recall : 0.8313 auroc : None

Updated Metrics (Majority Vote): f1 : 0.8172933469226564 accuracy : 0.8153 precision : 0.8242658318931743 recall : 0.8153 auroc : None
Original Metrics (Averaging Vectors): f1 : 0.850942076369477 accuracy : 0.8511 precision : 0.8533863986893996 recall : 0.8511 auroc : 0.9865711277777779
Updated Metrics (Averaging Vectors): f1 : 0.8378472535821491 accuracy : 0.8354 precision : 0.847339083480248 recall : 0.8354 auroc : 0.9851589166666667
Forgetting effectiveness (original): 0.02
Forgetting effectiveness (updated): 0.022

Simulation Question 3:
Membership inference attacks aim to determine whether a particular data sample was part of the training dataset of a machine learning model. This type of attack leverages differences in the model's behavior between training data and unseen data to infer membership status. Such attacks can lead to privacy breaches and reveal sensitive information about the training data.

Output Analysis:

Trained Model Scores:
Cross-validation of datasets 1 and 2 for the trained model: 49.50%
Cross-validation of datasets 3 and 4 for the unlearned model: 47.30%
The scores indicate that the trained model has a slightly higher than random chance of inferring membership, suggesting some overfitting or memorization of the training data.

Unlearned Model Scores:
Cross-validation of datasets 1 and 2 for the trained and unlearned model: 49.60%
Cross-validation of datasets 3 and 4 for the trained and unlearned model: 50.70%
These scores are close to 50%, indicating that the unlearned model does not significantly reveal whether a sample was part of the training data. This suggests that the unlearning process was effective in removing identifiable traces of the data points.

Evaluation:
The message "The unlearned model performed well, making it difficult to infer membership, indicating effective unlearning." confirms that the unlearning process successfully reduced the model's ability to distinguish between training and non-training data.

Add On. Simulation Question 1.:
The theoretical basis of implementation is as follows:
• **Backdoor Attack:**
A backdoor attack involves poisoning the training data so that the model learns to associate a specific trigger with a target label. During inference, the presence of this trigger in any input data will cause the model to misclassify it as the target label.

Output Analysis:

The outputs provide metrics for evaluating the model's performance on clean data and backdoor poisoned data. Specifically:

•Clean Metrics (Majority Vote): The accuracy of the model on clean test data using the majority vote method.
•Clean Metrics (Averaging Vectors): The accuracy of the model on clean test data using the averaging vectors method.
•Poisoned Metrics (Majority Vote): The accuracy of the model on backdoor poisoned test data using the majority vote method.
•Poisoned Metrics (Averaging Vectors): The accuracy of the model on backdoor poisoned test data using the averaging vectors method.
•Attack Success Rate (Majority Vote): The rate at which the backdoor attack successfully misclassified samples using the majority vote method.
•Attack Success Rate (Averaging Vectors): The rate at which the backdoor attack successfully misclassified samples using the averaging vectors method.
In this specific run, the clean accuracy is relatively low (39.2%), indicating that the model may need more tuning or additional training. The attack success rate (ASR) is almost identical to the clean accuracy, suggesting that the backdoor trigger is not significantly affecting the model's performance in this setup. This could imply either the backdoor is not effective or the model is not sufficiently trained to show a difference in behavior. Further adjustments and more epochs might be necessary for a clearer distinction.

Add On. Simulation Question 2.:

Output Analysis:

Clean Metrics (Majority Vote and Averaging Vectors): Both metrics indicate a clean accuracy of around 39.2%, showing the general performance of the model on non-poisoned data after unlearning. Poisoned Metrics (Majority Vote and Averaging Vectors): These metrics indicate the accuracy on poisoned data, which is slightly lower than the clean accuracy, suggesting that the model is not completely immune to backdoor attacks even after unlearning. Attack Success Rate (ASR): The ASR is about 39.14%, indicating that the backdoor attack is still moderately successful despite the unlearning effort. This shows the need for further refinement in the unlearning process to improve model robustness. The overall results suggest that while the SISA algorithm helps in selectively forgetting data, its effectiveness in mitigating backdoor attacks needs improvement, as indicated by the residual attack success rate.

Simulation Question 4: Baseline Model Training on CIFAR-10 Dataset

The theoretical basis of implementation is as follows:
The goal is to train a Convolutional Neural Network (CNN) on the CIFAR-10 dataset, which contains 60,000 32x32 color images in 10 classes, with 50,000 training images and 10,000 test images. The CNN consists of convolutional layers followed by fully connected layers. This approach is widely used in image classification tasks due to its ability to automatically learn hierarchical feature representations from raw pixel data.

Output Analysis:

Implementation:

- The CIFAR10Classifier model is defined with two convolutional layers, followed by dropout layers to prevent overfitting, and fully connected layers to produce the final output.
- The model is trained using the Adam optimizer and cross-entropy loss function for 10 epochs.
- The training data is divided into training and validation sets (80% and 20% respectively).

The implementation involves using the PyTorch framework to construct the CNN model, which includes defining the forward pass and setting up the data loaders for training and validation. The training loop iterates over the dataset, performing forward and backward passes, and updating the model parameters using the Adam optimizer. Validation accuracy is computed at the end of each epoch to monitor the model's performance on unseen data. **Output Analysis:**

- The model achieves a validation accuracy of 64.38% by the 10th epoch, indicating a decent performance for a baseline model on the CIFAR-10 dataset.
- The gradual increase in validation accuracy suggests that the model is learning effectively without overfitting.

The results indicate that the chosen model architecture and training parameters are appropriate for the task, providing a solid foundation for further experimentation with privacy-enhancing techniques.

Simulation Question 5: Training with Privacy Enhancements (Simple Noise Addition)

The theoretical basis of implementation is as follows:
Adding noise to gradients during training is a technique to enhance privacy by preventing overfitting to specific data points, making it harder for attackers to infer membership information. This approach is related to Differential Privacy, where the goal is to provide guarantees that the inclusion or exclusion of a single data point does not significantly affect the model's output.
In the first step, I used the Opacus library and the PrivateEngine function for implementing this part but due to many issues and the complexity of the input variables of this function, I decided to use a new method (adding noise to gradients) without using the Opacus library.

Output Analysis:

Implementation:

- A noise factor is added to the gradients during the backward pass to introduce randomness.

Specifically, Gaussian noise is sampled and added to the gradients of each model parameter.

- This technique aims to ensure that the model does not memorize specific details from the training data, thus enhancing privacy.

The implementation modifies the training loop to include noise addition after computing the gradients. By iterating through the model parameters and adding Gaussian noise to their gradients, the training process incorporates randomness, making it harder for an attacker to distinguish between members and non-members of the training set.
Output Analysis:

- The model with noise addition achieves a validation accuracy of 47.48%, which is lower than the baseline model.
- The added noise impacts the model's ability to learn effectively, resulting in decreased performance but potentially higher privacy.

The results show a trade-off between model accuracy and privacy. While the privacy-enhanced model performs worse in terms of accuracy, the added noise provides stronger privacy guarantees, as reflected in the reduced performance.

Simulation Question 6: Membership Inference Attacks and Privacy Evaluation
The theoretical basis of implementation is as follows:
Membership inference attacks aim to determine if a specific data point was part of the training dataset by analyzing the model's predictions. Evaluating the model's resilience to such attacks involves training attack models to classify data points as members or non-members. This type of attack exploits differences in the model's behavior on training versus non-training data.
As we discussed in the group of the course almost all of the students achieve almost the same accuracies for both Baseline and Privacy-Enhanced Models. I tried many different ways to implement this part but it didn't make two accuracies with sensible differences. In some methods that I've tried, I faced GPU memory errors in Kaggle! and it wasn't any solution for using these methods. All of these efforts resulted in the outputs below.

Output Analysis:

Implementation:

- Two models are trained: a baseline model and a privacy-enhanced model. The baseline model is trained without any privacy mechanisms, while the privacy-enhanced model uses noise addition during training.
- Attack models are trained to infer membership status based on the outputs of these models. These attack models are trained on shadow datasets to mimic the distribution of the target model's training data.

The implementation involves training the baseline and privacy-enhanced models as described in previous sections. Attack models are then trained using the outputs of these models on separate shadow datasets. These attack models learn to distinguish between member and non-member data points based on the target model's behavior.

Output Analysis:

- MIA Accuracy for Baseline Model: 80.0%
- MIA Accuracy for Privacy-Enhanced Model: 74.6%
- The privacy-enhanced model shows slightly better resilience against membership inference attacks.

The results indicate that the privacy-enhanced model offers marginally improved protection against membership inference attacks. However, the small difference suggests that further improvements in privacy techniques may be necessary to achieve significant gains in privacy.

Simulation Question 7: Improved Membership Inference Attacks

The theoretical basis of implementation is as follows:
Increasing the number of shadow models and improving the attack model's architecture can enhance the attacker's ability to infer membership, thus testing the model's privacy more rigorously. This approach leverages multiple shadow models to better capture the target model's decision boundaries. In this part, the key aspect is increasing the number of shadow models. The code has the following properties:
Training the Baseline Model and Privacy-Enhanced Model:
epochs: 10 for each model.
Number of models: 2 (one baseline and one privacy-enhanced).
Training Multiple Shadow Models:
numShadowModels: 5 shadow models.
epochs: 10 for each shadow model.
Training the Attacker Models:
epochs: 10 for each attacker model.
Number of attacker models: 2 (one for baseline, one for privacy-enhanced).

Output Analysis:

Implementation:

- Train multiple shadow models to better mimic the target model's behavior. These shadow models are trained on different subsets of the data to simulate the training process of the target model.
- Use these shadow models to generate more accurate attack data. The outputs of the shadow models are used to train the attacker models.
- Train the attacker model on this improved data. The attacker model is trained to distinguish between member and non-member data points based on the outputs of the shadow models.

The implementation involves training five shadow models to capture the behavior of the target model. These shadow models are then used to generate attack data, which is used to train the attacker models. This process enhances the attacker's ability to infer membership by leveraging a more comprehensive understanding of the target model's decision boundaries.
Output Analysis:

- Improved MIA Accuracy for Baseline Model: 99.994
- Improved MIA Accuracy for Privacy-Enhanced Model: 97.219

- The privacy-enhanced model still provides some protection, but the attack is much more effective due to the increased number of shadow models.

The results demonstrate that increasing the number of shadow models significantly enhances the attacker's ability to infer membership. While the privacy-enhanced model provides some resistance, the high attack accuracy highlights the need for more robust privacy-preserving techniques.

Simulation Question 8: Evaluating Different Attacker Models

The theoretical basis of implementation is as follows:
Evaluating different attacker models provides insight into how different model architectures impact the effectiveness of membership inference attacks. By comparing various attacker models, we can determine the robustness of the target model's privacy measures.
Before start implementing this parts code, I had saved my attacker models (torch.save). Then I modified the code that was provided for presentation such that works with my attacker model, also used the detailed for model under attack from Readme file.

Output Analysis:

Implementation:

- Use a simple binary classifier to evaluate the attack models. This classifier is trained to distinguish between member and non-member data points based on the target model's outputs.
- Load pre-trained attacker models and evaluate their performance on the new data. The attacker models are evaluated on separate test datasets to measure their effectiveness.

The implementation involves defining a binary classifier for the attacker model and training it on the outputs of the target model. Pre-trained attacker models are then loaded and evaluated on new data to assess their performance.
Output Analysis:

- Baseline Attacker Model:
  - Training Accuracy: 99.994%
  - High precision, recall, and F1 score, indicating the model is very effective in membership inference.
- Privacy-Enhanced Attacker Model:
  - Training Accuracy: 97.219%
  - Slightly lower precision, recall, and F1 score compared to the baseline, but still very effective.

The results show that different attacker models can significantly impact the effectiveness of membership inference attacks. The high accuracy of the baseline attacker model suggests that the target model's privacy measures are not entirely effective. The privacy-enhanced model offers better resistance, but further improvements are necessary to achieve robust privacy.