Introduction To Machine Learning Project: **All You Need is Privacy!**

Zahra Maleki & Hossein Anjidani

May 2024

1 Introduction

In this report, We will watch the first phase of the IML-2024 course which Dr. Amiri instructs. This project discusses privacy in ML which is crucial in real-life problems. These techniques are widely usable, especially in distributed learning. Here is the sight of these phase:

1.1 Summary

1.1.1 Project Overview

- **Course:** Introduction to Machine Learning (25737-2)
- **Semster:** Spring 1402-03
- **University:** Sharif University of Technology
- **Instructor:** Dr. R. Amiri
- **Due Date:** Khordad 11, 1403 at 23:55

1.1.2 Key Focus

The project addresses the critical issue of privacy in machine learning, with an emphasis on Machine Unlearning and Private Training Models.

1.1.3 Machine Unlearning:

- **Objective:** To enable a machine learning model to forget specific data points without retraining from scratch.
- **Importance:** Compliance with privacy regulations like GDPR, error correction, and data management.
- **SISA Algorithm :** A structured approach for efficient machine unlearning, involving sharding, isolation, slicing, and aggregation.

1.1.4 Private Training Models:

- **Purpose:** To protect sensitive data during model training and safeguard against Membership Inference Attacks.
- **Techniques:** Differentially private training, regularization, normalization temperature, and adding noise for privacy.

1.1.5 Membership Inference Attack:

- **Concern:** Determining if a data record was part of a model's training dataset1011.
- **Types:** White-box, gray-box, and black-box attacks.
- **Method:** Training an attack model to classify whether data was in the training set, using shadow models for training.

2 Machine Unlearning

2.1 Theory Question 1:

1. Majority Voting: Each constituent model votes for a class label, and the majority label is output. Strength is simplicity, it effectively averages out errors. Weakness is that it only uses hard class labels rather than probabilities, and accurate constituents may be overwhelmed by many inaccurate ones.
2. Averaging Probabilities: Each constituent outputs a probability distribution over classes, and the average distribution is output. Strength is it makes use of probability information rather than just labels. Weakness is that it assumes constituents are reasonably well-calibrated, and accuracy may decrease if some constituents are very inaccurate.
3. Stacked Generalization: Train a meta-learner on the outputs of the constituent models. Strength is the meta-learner can learn the best way to combine the constituents based on their actual performance, mitigating weaknesses of simple averaging. Weakness is increased complexity, and may overfit if not regularized properly given the small number of constituent models.

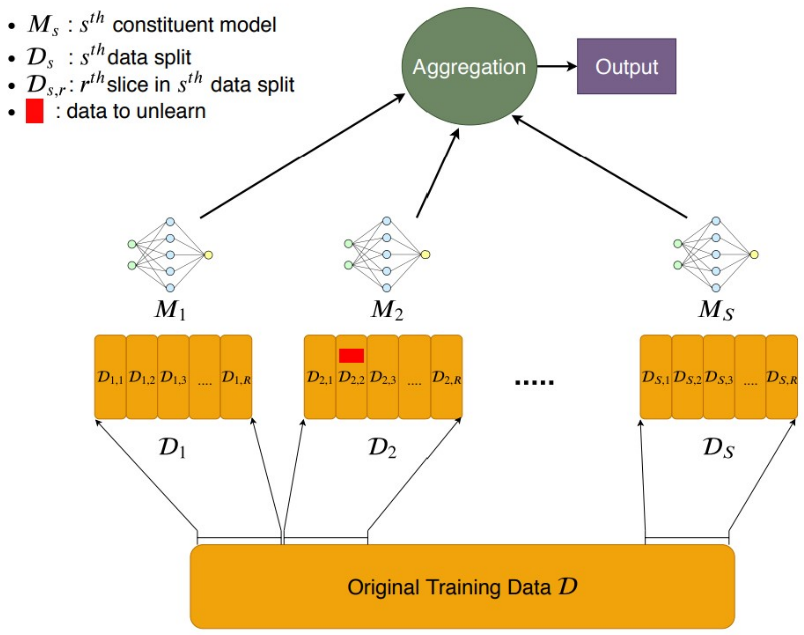
The paper found that for simpler learning tasks on datasets like Purchase and SVHN, both majority voting and averaging probabilities worked reasonably well with SISA training and did not significantly degrade accuracy compared to the baseline.

2.2 Theory Question 2:

This method performs on different scales of data and models, and any significant degradation has not been observed in efficiency even for more complex tasks like ImageNet classification. However, there are a few scenarios where the SISA training method could become inefficient.

1. High volume of unlearning requests: If the number of unlearning requests is very large, approaching the total number of data points, it may become inefficient to keep retraining shards and redistributing data. To mitigate this, the paper suggests batching unlearning requests to minimize the number of retrainings needed.
2. Data that is highly correlated/non-IID: If the data is not independently and identically distributed across shards, it may be ineffective to train constituents in isolation. The paper proposes taking the data distribution into account when assigning points to shards, to minimize cross-shard correlation.
3. Very large models: For models with millions of parameters, the overhead of maintaining multiple constituent copies could become prohibitive, even if trained independently. One approach could be hierarchical sharding, where lower layers are shared across constituents to limit replication.

It seems to me that having a correlated dataset could pose a more challenging issue for this method.



M_s : s^{th} constituent model
$\mathcal{D}_s$: s^{th} data split
$\mathcal{D}_{s,r}$: r^{th} slice in s^{th} data split
🟥 : data to unlearn
Aggregation
Output
M_1
M_2
M_S
$\mathcal{D}_{1,1}$ $\mathcal{D}_{1,2}$ $\mathcal{D}_{1,3}$ .... $\mathcal{D}_{1,R}$
$\mathcal{D}_{2,1}$ $\mathcal{D}_{2,2}$ $\mathcal{D}_{2,3}$ .... $\mathcal{D}_{2,R}$
$\mathcal{D}_{S,1}$ $\mathcal{D}_{S,2}$ $\mathcal{D}_{S,3}$ .... $\mathcal{D}_{S,R}$
$\mathcal{D}_1$
$\mathcal{D}_2$
$\mathcal{D}_S$
Original Training Data $\mathcal{D}$

Figure 1: When data needs to be unlearned, only one of the constituent models whose shards contains the point to be unlearned needs to be retrained— retraining can start from the last parameter values saved before including the slice containing the data point to be unlearned.

2.3 Theory Question 3:

There are a few key metrics that would be important to evaluate the performance of the SISA machine unlearning algorithm.

1. Accuracy: The most important metric is the accuracy of the model after portions of the training data have been unlearned. The goal of SISA is to allow for unlearning with minimal impact on overall model accuracy. So accuracy on a test/holdout set would need to be measured and compared to a baseline model trained on all data without any unlearning. Small losses in accuracy would indicate the algorithm is successful.

2. Unlearning time: As reducing computational cost/time is a key goal of SISA, the time taken to perform the unlearning operations would need to be thoroughly tested. Unlearning time under SISA would need to be demonstrably lower than retraining from scratch (the baseline approach). Tests over varying numbers of data points to unlearn could show how SISA scales.

3. Isolation of influence: Another critical metric is how well the unlearning isolates the influence of removed data points just to the targeted shards/slices, without affecting other areas that were not unlearned. Tests could check if unlearning specific points only impacts accuracy on similar future points and not others. This helps show the guarantee of true ”forgetting”.

Together, evaluating these metrics on different experimental conditions would best establish the effectiveness and validity of the SISA algorithm for machine unlearning.

2.4 Theory Question 4:

The models’ architecture has an effect on both the learning and unlearning time as well as the performance during learning and unlearning.while complex models increase SISA’s learning and unlearning times, techniques like transfer learning can help tradeoff performance vs time. The paper also demonstrated SISA is effective across different model complexities from simple to deep networks.

- Complex architectures like deep neural networks (DNNs) generally require more training time compared to simpler models. This is because DNNs involve learning a large number of parameters across multiple layers.
- Similarly, retraining or unlearning a DNN would also take more time than simpler models due to the size of the model. The SISA paper observed speedups were lower for complex ImageNet tasks compared to simpler datasets.
- Techniques like transfer learning could help reduce unlearning performance degradation for complex models/datasets under SISA. Transferring knowledge from publicly trained models acted as a regularizer.
- Deeper networks have higher capacity to memorize the training data, making true unlearning/forgetting potentially more difficult compared to shallow architectures. SISA aims to address this.
- More parameters also mean the model is more diffuse, making isolating the effect of removed points challenging. SISA's sharding and slicing helps mitigate such issues.
- Architectures involving recurrence/feedback could slow down SISA's unlearning since influence may propagate across time steps.

3 Private Training Models

3.1 Theory Question 5:

Machine learning algorithms learn from data, adjusting their model parameters to capture patterns and relationships. However, we want these models to generalize well without memorizing specific details about individual training examples. For instance, when building a cancer diagnosis model, we don't want it to reveal sensitive information about patients' medical histories inadvertently.

The challenge lies in achieving both accurate model performance and privacy. Generalization (good performance on test examples) doesn't automatically imply privacy. Privacy must be ensured for everyone, including outliers, which may deviate from the assumed data distribution1.

3.1.1 Differentially Private Algorithms:

1. **Differential Privacy (DP):**
   - DP provides a rigorous privacy guarantee by quantifying the impact of an individual's data on the model's output.
   - It ensures that the model's behavior doesn't significantly change when a single data point is added or removed.
   - The core idea is to inject controlled noise into the learning process to protect individual privacy.
2. **DP-SGD (Stochastic Gradient Descent):**
   - DP-SGD is a widely used approach for training deep neural networks with differential privacy.
   - It adds noise to each gradient computed during SGD (Stochastic Gradient Descent) updates.
   - The noise level is controlled by a privacy parameter (often denoted as ϵ).
   - By adjusting ϵ, we balance privacy and utility (model accuracy).

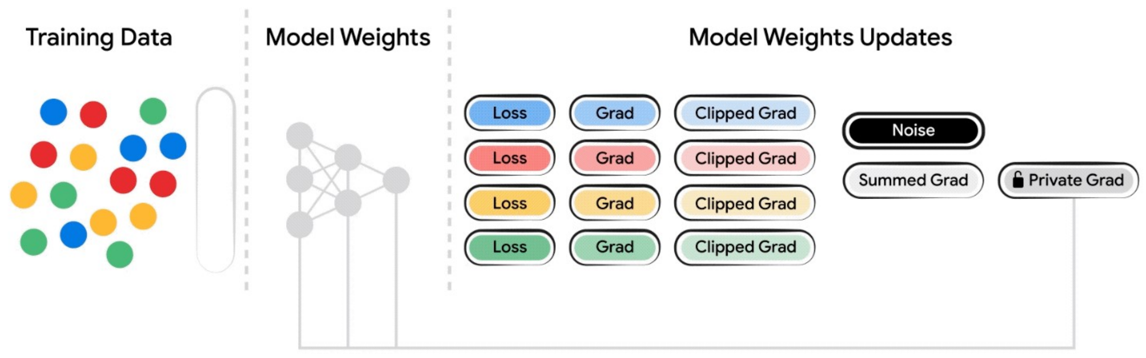


Figure 2: This pictures shows how to update the weights by using a function of the inputs and their gradients without the specific weight and also adding noise(DP-SGD) for more privacy. click here for complete GIF

3. **Composition Attacks:**
    - Traditional privacy approaches (e.g., k-anonymity, l-diversity) are vulnerable to composition attacks.
    - Composition attacks exploit auxiliary information to violate privacy protection.
    - For example, researchers de-anonymized Netflix Prize participants using IMDb data and movie ratings.

3.1.2 Techniques for Training Differentially Private Models:

1. **Input Data Level:**
    - Introduce privacy at the input data level.
    - Perturb the data before feeding it into the model.
    - Examples include adding noise to features or applying transformations.
2. **During Training:**
    - DP-SGD: Add noise to gradients during each iteration(Figure 1).
    - Other techniques include DP-Adam (a variant of Adam optimizer), DP-FedAvg (for federated learning), and DP-AdaFEST (for large embedding models).
3. **At Inference:**
    - Introduce privacy during prediction.
    - Post-process model outputs to ensure privacy.
    - However, this is the weakest form of privacy protection.

Remember that achieving privacy is a delicate balance. We want models to perform well while safeguarding individual privacy. Differential privacy provides a principled framework to achieve this balance

3.2 Theory Question 6:

3.2.1 Regularization Techniques:

1. **What is Overfitting?**
   - Overfitting occurs when a model becomes too complex and starts memorizing noise and randomness in the training data.
   - It fits the training data too closely, leading to poor generalization on unseen data.
2. **Role of Regularization:**
   - Regularization aims to prevent overfitting by adding a penalty term to the model's objective function during training.
   - The penalty discourages the model from becoming overly complex.
3. **Common Regularization Techniques:**
   - **L2 Regularization (Ridge Regression):**
     - Adds the sum of squared weights (L2 norm) to the loss function.
     - Encourages smaller weight values, preventing extreme parameter values.
     - Helps the model generalize better.
   - **L1 Regularization (Lasso Regression):**
     - Adds the sum of absolute weights (L1 norm) to the loss function.
     - Promotes sparsity by driving some weights to exactly zero.
     - Useful for feature selection.
   - **Elastic Net Regularization:**
     - Combines L1 and L2 regularization.
     - Balances the benefits of both techniques.

3.2.2 Normalization Techniques:

1. **Why Normalize Data?**
   - Data often have different scales (e.g., age vs. income).
   - Normalization ensures that features have similar ranges, aiding model convergence and performance.
2. **Common Normalization Techniques:**
   - **Min-Max Scaling (Normalization):**
     - Scales features to a specified range (e.g., [0, 1]).
     - Formula: $X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$
   - **Z-Score (Standardization):**
     - Standardizes features to have zero mean and unit variance.
     - Formula: $X_{standardaized} = \frac{X - \mu}{\sigma}$

3.2.3 Relationship to Differential Privacy:

- **Adding Noise in Differential Privacy:**
  - Differential privacy injects controlled noise into model training to
  - protect individual privacy.
  - The noise level is controlled by a privacy parameter (often denoted as ϵ).
  - It balances privacy and utility (model accuracy).
- **Regularization and Noise:**
  - While regularization and differential privacy both aim to prevent overfitting, they operate differently.
  - Regularization adjusts the model's loss function by penalizing complexity(Adding a special calculatable term).
  - Differential privacy adds noise directly to gradients during training iterations.

In summary, regularization and normalization techniques help improve model performance and prevent overfitting, while differential privacy focuses on privacy protection by adding noise. Although they serve distinct purposes, all three contribute to building robust and generalizable models.

3.3 Theory Question 7:

1. **Ensemble Model:**
   - Use independent models for independent features and then use an aggregation method (as We saw in machine unlearning) to make the prediction reverse function much more complicated.
   - makes predictions of different models trained on some or all features of the dataset.
2. **Transfer Learning:**
   - Leveraging pre-trained models (usually on large datasets) and fine-tuning them for specific tasks. Due to larger previously seen data; predicting a part of the dataset is harder in comparison with a trained model with limited data.
   - Transfer learning accelerates training and improves performance, especially when you have limited labeled data.
3. **Feature Engineering:**
   - Creating new features from existing ones to enhance model performance. Note that the generation function should not be random but it should be enough complicated.
   - Techniques include polynomial features, interaction terms, and domain-specific transformations.

4 Membership Inference Attack

4.1 Theory Question 8 & 9:

4.1.1 Model-based synthesis:

If the attacker does not have real training data nor any statistics about its distribution, he can generate synthetic training data for the shadow models using the target model itself. The intuition is that records that are classified by the target model with high confidence should be statistically similar to the target's training dataset and thus provide good fodder for shadow models. The

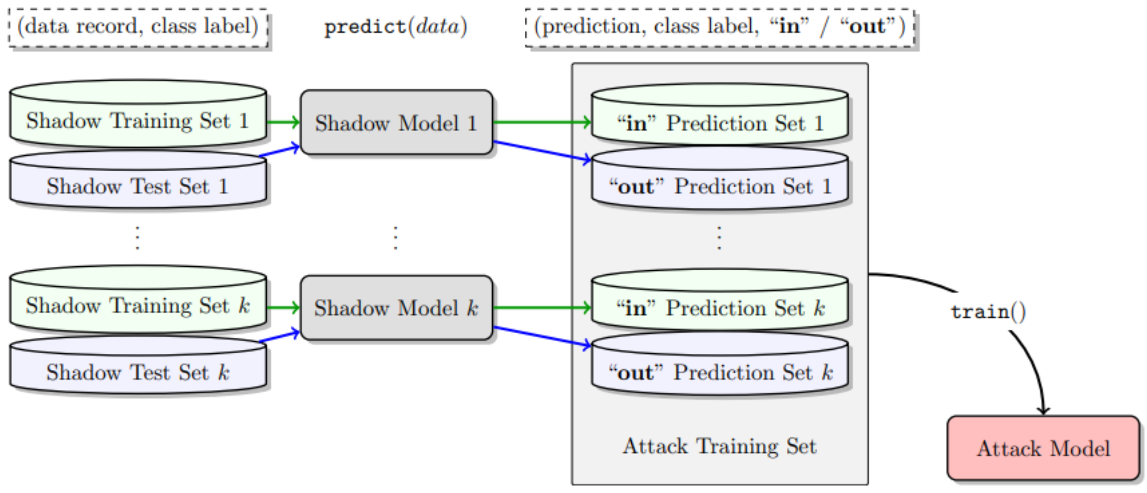


Figure 3: Training the attack model on the inputs and outputs of the shadow models. For all records in the training dataset of a shadow model, we query the model and obtain the output. These output vectors are labeled "in" and added to the attack model's training dataset. We also query the shadow model with a test dataset disjoint from its training dataset. The outputs on this set are labeled "out" and also added to the attack model's training dataset. Having constructed a dataset that reflects the black-box behavior of the shadow models on their training and test datasets, we train a collection of c_{target} attack models, one per each output class of the target model

synthesis process runs in two phases: (1) search, using a hill-climbing algorithm, the space of possible data records to find inputs that are classified by the target model with high confidence; (2) sample synthetic data from these records. After this process synthesizes a record, the attacker can repeat it until the training dataset for shadow models is full.

4.1.2 Statistics-based synthesis:

The attacker may have some statistical information about the population from which the target model's training data was drawn. For example, the attacker may have prior knowledge of the marginal distributions of different features. In our experiments, we generate synthetic training records for the shadow models by independently sampling the value of each feature from its own marginal distribution. The resulting attack models are very effective.

4.1.3 Noisy real data:

The attacker may have access to some data that is similar to the target model's training data and can be considered as a "noisy" version thereof. In our experiments with location datasets, we simulate this by flipping the (binary) values of 10% or 20% randomly selected features, then training our shadow models on the resulting noisy dataset. This scenario models the case where the training data for the target and shadow models are not sampled from exactly the same population, or else sampled in a non-uniform way.

4.2 Theory Question 10

The main idea behind our shadow training technique is that similar models trained on relatively similar data records using the same service behaves in a similar way. This observation is empirically borne out by our experiments in the rest of this paper. Our results show that learning how to infer membership in shadow models' training datasets (for which we know the ground truth and can easily compute the cost function during supervised training) produces an attack model that successfully infers membership in the target model's training dataset, too.

We query each shadow model with its own training dataset and with a disjoint test set of the same size. The outputs on the training dataset are labeled “in,” and the rest are labeled “out.” Now, the attacker has a dataset of records, the corresponding outputs of the shadow models, and the in/out labels. The objective of the attack model is to infer the labels from the records and corresponding outputs.
Figure 2 shows how to train the attack model. For all $(x, y) \in D^{train}_{shadow}$, compute the prediction vector $y = f^{train}_{shadow}(x)$ and add the record (y, y, in) to the attack training set D^{train}_{attack}. Let D^{test}_{shadow} be a set of records disjoint from the training set of the ith shadow model. Then, for all $(x, y) \in D^{test}_{shadow}$ compute the prediction vector $y = f^{i}_{shadow}(x)$ and add the record (y, y, out) to the attack training set D^{train}_{attack}. Finally, split D^{train}_{attack} attack into c_{target} partitions, each associated with a different class label. For each label y, train a separate model that, given y, predicts the in or out membership status for x.
If we use model-based synthesis from Section V-C, all of the raw training data for the attack model is drawn from the records that are classified by the target model with high confidence. This is true, however, both for the records used in the shadow models' training datasets and for the test records left out of these datasets. Therefore, it is not the case that the attack model simply learns to recognize inputs that are classified with high confidence. Instead, it learns to perform a much subtler task: how to distinguish between the training inputs classified with high confidence and other, non-training inputs that are also classified with high confidence.
In effect, we convert the problem of recognizing the complex relationship between members of the training dataset and the model's output into a binary classification problem. Binary classification is a standard machine learning task, thus we can use any state-of-the-art machine learning framework or service to build the attack model. Our approach is independent of the specific method used for attack model training. For example, in Section VI we construct the attack model using neural networks and also using the same black-box Google Prediction API that we are attacking, in which case we have no control over the model structure, model parameters, or training metaparameters but still obtain a working attack model.

4.3 Theory Question 11:

1. **Effect of Shadow Training Data Generation Methods:**
   - The method used to generate shadow training data significantly impacts the attack model's accuracy.
   - **Paired Shadow and Non-Shadow Regions:**
     - If paired shadow and non-shadow regions are used, the attack model learns from direct correspondence between shadow and non-shadow examples.
     - Accuracy may be high when the shadow generation process is consistent and the paired data is diverse.
   - **Stylized Shadow Generation:**
     - This method relies on stylizing non-shadow regions to resemble actual shadows.
     - Accuracy depends on the quality of stylization; if it closely mimics real shadows, the attack model performs well.
   - **Black-Box Access and Adversarial Data:**
     - Constructing adversarial datasets using shadow models can lead to accurate attack models.
     - The choice of shadow models and adversarial data quality matters.
2. **Effect of Number of Classes and Training Data per Class**
   - **Number of Classes:**
     - More classes increase the complexity of the attack task.
     - Accuracy tends to decrease as the number of classes grows.

    - Multi-class attacks require more data and better feature representations.
  - **Training Data per Class:**
    - Adequate training data per class is crucial.
    - Insufficient data leads to poor generalization and overfitting.
    - Too much data can also cause overfitting if the model memorizes noise.

3. **Effect of Overfitting:**
   - Overfitting occurs when the attack model learns noise or specific details from the training data.
   - **Impact on Accuracy:**
     - Overfitting reduces the attack model's accuracy on unseen data.
     - It performs well on the training data but poorly on the test data.
   - **Preventing Overfitting**
     - Regularization techniques (L1, L2) help prevent overfitting.
     - Early stopping, dropout, and cross-validation are also effective.
     - Balancing model complexity and data size is essential.

Remember that achieving a balance between model complexity, data quality, and generalization is crucial for accurate attack models.

5 Refrences

First of all, anything We know is based on what our instructor, Dr. Amiri taught us so the main source is IML slides. but We use some site and paper to answer this question which will be mentioned below:

1. Navigating Overfitting: Understanding and Implementing Regularization Techniques in Data Science
2. What is the difference between normalization and regularisation in machine learning
3. Regularization Techniques in Machine Learning
4. Regularization in Machine Learning (with Code Examples)
5. What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling
6. Differentially Private Decentralized Deep Learning with Consensus Algorithms
7. Learning Differentially Private Mechanisms
8. Sparsity-Preserving Differentially Private Training of Large Embedding Models
9. Making ML models differentially private: Best practices and open challenges
10. How to deploy machine learning with differential privacy
11. Membership Inference Attacks Against Machine Learning Models
12. University of Toronto , Vector Institute , University of Wisconsin-Madison,Machine Unlearning