Introduction to Machine Learning

AUTHORS

Matin Mirzababaie - 400102114
Ali Ekhterachian - 400100576

June 25, 2024

Contents

1 Machine Unlearning

1.1 Theory Question 1

For the SISA training approach, three aggregation methods are proposed: Majority Voting, Weighted Voting, and Confidence-Based Aggregation. Each method has its own strengths and weaknesses, which are detailed below.

1.1.1 Majority Voting

How it works: Each model makes a prediction, and the final decision is based on the most common prediction among all models.

Pros:

- **Simple**: Easy to implement and understand.
- **Resilient**: Can handle errors from individual models well, as the majority decision can counteract incorrect predictions from a few models.
- **No Need for Model Calibration**: Doesn't rely on the confidence levels of the models, simplifying implementation.

Cons:

- **Class Imbalance Issues**: Can favor predictions from classes that are more commonly predicted by individual models.
- **Ignores Confidence Levels**: Does not consider how confident each model is in its prediction, which might overlook useful information.

1.1.2 Weighted Voting

How it works: Each model's prediction is weighted based on its accuracy or other performance metrics. The final decision is based on the sum of these weighted predictions.

Pros:

- **Performance-Aware**: Gives more influence to better-performing models, potentially improving overall accuracy.

- **Adjustable**: Can be fine-tuned based on various performance indicators or specific criteria.

Cons:

- **More Complex**: Requires additional computation to determine and apply weights based on model performance.
- **Risk of Overfitting**: Weights need careful adjustment to avoid overfitting to the validation data used to determine them.

1.1.3 Confidence-Based Aggregation

How it works: Each model provides a probability distribution over all possible classes. These distributions are averaged, and the class with the highest average probability is chosen as the final prediction.

Pros:

- **Comprehensive Use of Information**: Utilizes full probability distributions, which can lead to more informed and accurate final decisions.
- **Confidence Consideration**: Takes into account the certainty of each model's predictions, which can improve decision-making.

Cons:

- **Computationally Demanding**: Requires more computation to average probability distributions.
- **Calibration Needed**: Models must be well-calibrated to ensure their probability distributions are accurate, adding complexity to the process.

In summary:

- **Majority Voting** is straightforward and robust but may miss out on confidence information and can be skewed by class imbalances.
- **Weighted Voting** leverages model performance but is more complex and can overfit if not managed carefully.
- **Confidence-Based Aggregation** makes the most informed decisions by considering model confidence but requires well-calibrated models and more computational power.

1.2 Theory Question 2

High Fragmentation

- **Issue**: If the dataset is fragmented into too many shards and slices, managing and aggregating models can become overly complex and resource-intensive.
- **Mitigation**: Optimize shard and slice sizes to balance granularity with manageability. Use cross-validation to determine the best partitioning strategy that minimizes overhead while achieving effective unlearning.

Frequent Unlearning Requests

- **Issue**: In environments with frequent unlearning requests, the constant retraining of models for affected shards and slices can be computationally expensive.
- **Mitigation**: Implement batch processing for unlearning requests to reduce overhead. Process multiple requests together periodically instead of handling each request immediately.

Interdependent Data

- **Issue**: Dependencies among data points (e.g., time-series data, graphs) can complicate the unlearning process because removing one data point may require re-evaluating a larger portion of the dataset.
- **Mitigation**: Use advanced partitioning techniques that account for data dependencies. Consider adaptive partitioning strategies tailored to specific data characteristics.

Large Shard Size

- **Issue**: If shards are too large, the benefits of isolated retraining diminish, and the process may become as computationally intensive as retraining the entire model.
- **Mitigation**: Keep shard sizes manageable by regularly reviewing and adjusting the partitioning strategy. Consider dynamic sharding based on workload and data characteristics.

Complex Aggregation

- **Issue**: Aggregating models trained on different shards and slices can be complex, especially if models vary significantly in their learned parameters.
- **Mitigation**: Use ensemble learning techniques for efficient aggregation. Develop robust algorithms for weighted averaging or stacking to integrate models effectively.

Recommendations

- **Monitor and Adapt**: Continuously monitor performance metrics such as retraining times and computational load. Adapt partitioning strategies based on observed performance.
- **Hybrid Approaches**: Combine SISA with other unlearning techniques where appropriate to optimize efficiency.
- **Algorithmic Improvements**: Invest in research to develop more efficient retraining algorithms suited for incremental updates and diverse datasets.
- **Resource Management**: Allocate sufficient computational resources and consider scalable solutions for handling varying workloads effectively.

1.3 Theory Question 3

Accuracy

Measures how effectively the algorithm maintains or improves the model's overall accuracy after unlearning specific information. It assesses the algorithm's ability to mitigate negative impacts on model performance.

Retention Rate

Quantifies the proportion of retained knowledge or model performance (e.g., accuracy, precision, recall) after unlearning. A higher retention rate indicates better preservation of the original model's capabilities.

Training Time

Evaluates the time required for the unlearning process, crucial in real-time or dynamic environments. A good algorithm should be efficient and minimize additional retraining time.

Memory Usage

Assesses the additional memory required during unlearning and subsequent retraining. It's important in resource-constrained environments to manage memory efficiently.

Robustness to Change

Measures how well the algorithm adapts to changes in the dataset or model structure over time. It reflects the algorithm's ability to maintain effectiveness under evolving conditions.

Privacy Preservation

For privacy-sensitive applications, evaluates how effectively the algorithm removes sensitive or personal data from the model.

Reasoning Behind the Choice

The selection of metrics should align closely with the specific objectives and requirements of the application using the unlearning algorithm. For instance:

- **Accuracy and Retention Rate**: Crucial for maintaining model performance.
- **Training Time and Memory Usage**: Essential for efficiency in real-time scenarios.
- **Robustness and Privacy Preservation**: Important for adapting to changes and ensuring data protection.

Choosing appropriate metrics ensures stakeholders can effectively evaluate the unlearning algorithm based on their specific needs, whether focused on performance, efficiency, adaptability, or privacy considerations.

1.4 Theory Question 4

Learning Phase with SISA

- **Time Efficiency**:
  - **Influence of Complexity**: Model architecture complexity affects initial training time. More intricate architectures require more computational resources and time to converge.
  - **Incremental Updates**: SISA's approach of incremental updates can streamline learning by focusing adjustments on relevant data subsets, potentially reducing re-training needs.
- **Performance**:
  - **Impact of Unlearning**: After employing SISA for unlearning, architecture adaptability dictates how well the model sustains or enhances its performance. Effective unlearning should enhance subsequent task performance by eliminating obsolete knowledge.
  - **Managing Re-learning**: Complex architectures may require careful management during re-learning to restore optimal performance post-unlearning. SISA's incremental updates mitigate performance decline by focusing adjustments on pertinent model components.

Specific Considerations with SISA

- **Incremental Methodology**: SISA's incremental parameter adjustments optimize learning and unlearning phases by prioritizing relevant updates.
- **Algorithmic Efficiency**: The efficiency of SISA algorithms in handling model updates and adjustments is crucial for minimizing time overhead during learning and unlearning operations.

2 Private Training Models

2.1 Theory Question 5

2.1.1 Differentially Private Algorithms

Differentially private algorithms are designed to protect the privacy of individual data points while allowing useful information to be extracted from datasets. Here's a human-like explanation of how they work and their techniques for training a differentially private model:

Imagine you have a bunch of sensitive information about people, like their medical records or spending habits, and you want to learn insights from this data without compromising anyone's privacy. That's where differentially private algorithms come in.

How They Work: Differentially private algorithms add a layer of protection by injecting noise or randomness into the computations done on the data. This noise is carefully calibrated so that the statistical results drawn from the data remain accurate for groups of people, but it becomes impossible to determine the contribution of any single individual with certainty.

Techniques for Training: When training a differentially private model, there are a few key techniques used:

1. **Noise Addition:** During the training process, noise is added to the gradients (which indicate the direction of improvement for the model) or to the model parameters themselves. This ensures that the updates made during training don't reveal too much about any single individual's data.

2. **Privacy Budget:** Think of this as a limit on how much privacy you're willing to trade off for accuracy. By setting a privacy budget, you control how much noise is added to each computation or how much information is allowed to leak about individual data points.

3. **Aggregation Techniques:** Sometimes, instead of adding noise directly to each computation, data from multiple individuals or sources is aggregated first. This pooling of information helps to obscure the contributions of any single individual, providing stronger privacy guarantees.

4. **Objective Function Modification:** Adjusting the loss function or objective function of the model can also enhance privacy. By incorporating penalties for learning specific information or by limiting the influence of individual data points, the model can learn general trends without overfitting to any particular data.

In essence, differentially private algorithms balance the need for accurate insights with the protection of individual privacy. They ensure that even if someone tries to analyze the output, they won't be able to tell much about any single person in the dataset. It's like keeping everyone's secrets safe while still learning valuable lessons from the group as a whole.

2.2 Theory Question 6

2.2.1 Regularization

Overview of Regularization Regularization techniques like L1 and L2 help prevent a model from becoming too complex during training. By adding penalties to the model's learning process, these techniques encourage it to focus on important patterns rather than memorizing exact details from the training data. In the context of SISA machine unlearning algorithms, regularization is crucial to ensure that the model retains its utility while avoiding overfitting to sensitive data, thus maintaining privacy.

Application in SISA Machine Unlearning In SISA machine unlearning algorithms, regularization plays a vital role in enhancing model robustness. It helps in controlling the complexity of the model, which is essential for protecting sensitive information. By promoting simpler and more generalizable models, regularization indirectly contributes to preserving privacy by preventing the model from memorizing specific details about sensitive data points.

2.2.2 Normalization

Overview of Normalization Normalization involves scaling data to a standard range, such as adjusting features so they have similar scales or distributions. In SISA machine unlearning algorithms, normalization ensures that the model learns from data efficiently and without bias towards certain features. This process helps the model understand patterns better without being skewed by differences in data scales, thereby contributing to both model performance and privacy preservation.

Implementation in SISA Machine Unlearning In the implementation of SISA machine unlearning algorithms, normalization is essential for ensuring that the model processes data uniformly. By standardizing the input features, normalization aids in preventing the model from inadvertently focusing on irrelevant or non-representative aspects of the data. This improves the overall effectiveness of the model while safeguarding privacy by maintaining consistent data handling practices.

2.2.3 Relation to Adding Noise in Differential Privacy

Comparison with Adding Noise Adding noise in differential privacy is a fundamental technique to prevent the model's output from revealing specific details about individual data points. It involves adding carefully calculated noise during computations to protect sensitive information. While regularization and normalization are essential for improving model robustness and generalization, they do not provide the rigorous privacy guarantees of differential privacy through noise addition.

Integration with Differential Privacy Principles Despite not directly providing differential privacy guarantees, regularization and normalization in SISA machine unlearning algorithms align with the principles of differential privacy. They promote practices that mitigate the risk of overfitting and

ensure that the model learns generalizable patterns without compromising individual data privacy. By incorporating these techniques, SISA machine unlearning algorithms can enhance privacy protection while maintaining model performance and utility.

2.3 Theory Question 7

2.3.1 Federated Learning

Description Federated Learning enables training models across decentralized devices (such as smartphones or edge devices) without transferring raw data to a central server. Instead, only model updates (gradients) are sent to the server, preserving user privacy.

Application This technique is particularly useful in scenarios where data cannot leave its source due to privacy concerns or regulatory restrictions.

2.3.2 Homomorphic Encryption

Description Homomorphic encryption allows computations to be performed on encrypted data without decrypting it. This enables training models on encrypted data while preserving data privacy.

Application It is used in situations where data privacy is crucial, such as healthcare or financial data analysis.

2.3.3 Secure Multi-Party Computation (MPC)

Description MPC allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. Each party holds a private input and collaborates to compute the model without revealing their data.

Application MPC is useful when multiple parties need to collaborate on model training while keeping their individual data confidential.

2.3.4 Differential Privacy Mechanisms

Description Beyond adding noise, differential privacy includes techniques like data perturbation, query restriction, and output perturbation. These methods ensure that statistical queries or outputs of machine learning models do not reveal sensitive information about individual data points.

Application Differential privacy is widely used in scenarios requiring strong privacy guarantees, such as government statistics, social sciences, and healthcare.

2.3.5 Privacy-Preserving Data Synthesis

Description Generating synthetic data that closely resembles original data while protecting individual identities. This synthetic data can then be used for training models without exposing real individual information.

Application Useful in situations where access to real data is restricted due to privacy concerns but synthetic data can adequately represent the original dataset's statistical properties.

2.3.6 Model Distillation

Description Training a smaller, less complex model to mimic the behavior of a larger, more complex model trained on sensitive data. This allows for preserving the knowledge encoded in the original model without exposing sensitive details.

Application Effective when deploying models in sensitive environments or applications where model size and computational resources are limited.

2.3.7 Data Augmentation with Privacy in Mind

Description Augmenting training data by applying transformations or perturbations that preserve the statistical properties of the data while obfuscating sensitive information.

Application Enhances the robustness and privacy of models trained on augmented data, making it harder to infer sensitive information from the model's outputs.

3 Membership Inference Attack

3.1 Theory Question 8

3.1.1 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 The 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.
- The pseudocode for this case is found in Algorithm 1 within the article.
- This synthesis procedure works only if the adversary can efficiently explore the space of possible inputs and discover inputs that are classified by the target model with high confidence. For example, it may not work if the inputs are high resolution images and the target model performs a complex image classification task.

3.1.2 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.

3.1.3 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.

3.2 Theory Question 9

In section 2, we discussed this approach and highlighted its weaknesses. Now, let's delve into its algorithm.

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**
    $\mathbf{y} \leftarrow f_{target}(\mathbf{x})$ ▷ query the target model
    if $\mathbf{y}_c \geq y_c^*$ **then** ▷ accept the record
      if $y_c > conf_{min}$ 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
        $k \leftarrow \max(k_{min}, \lfloor k/2 \rfloor)$
        $j \leftarrow 0$
      end if
    end if
    $\mathbf{x} \leftarrow$ RANDRECORD($\mathbf{x}^*, k$) ▷ randomize k features
  end for
  return $\perp$ ▷ failed to synthesize
end procedure

- First, fix class c for which the attacker wants to generate synthetic data. The first phase is an iterative process. Start by randomly initializing a data record x. Assuming that the attacker knows only the syntactic format of data records, sample the value for each feature uniformly at random from among all possible values of that feature. In each iteration, propose a new record. A proposed record is accepted only if it increases the hill-climbing objective: the probability of being classified by the target model as class c.

- Each iteration involves proposing a new candidate record by changing k randomly selected features of the latest accepted record x^*. This is done by flipping binary features or resampling new values for features of other types. We initialize k to k_{max} and divide it by 2 when rej_{max} subsequent proposals are rejected. This controls the diameter of search around the accepted record in order to propose a new record. We set the minimum value of k to k_{min}. This controls the speed of the search for new records with a potentially higher classification probability y_c.

- The second sampling phase starts when the target model's probability y_c that the proposed data record is classified as belonging to class c is larger than the probabilities for all other classes and also larger than a threshold confmin. This ensures that the predicted label for the record is c, and that the target model is sufficiently confident in its label prediction. We select such a record for the synthetic dataset with probability y_c, and if selection fails, repeat until a record is selected.

3.3 Theory Question 10

- 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" 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.

- If we use model-based synthesis 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.

3.4 Theory Question 11

3.4.1 Effect of the Shadow Training Data Generated Using Different Methods

- **Noisy Data:** As written in the article, precision of the attacks trained on the shadow models whose training datasets are noisy versions of the real data (disjoint from the target model's training dataset but sampled from the same population). Precision drops as the amount of noise increases, but the attack still outperforms the baseline and, even with 10% of the features in the shadows' training data replaced by random values, matches the original attack. This demonstrates that our attacks are robust even if the attacker's assumptions about the distribution of the target model's training data are not very accurate.

- **Comparison of Precision**: When shadow models are trained on real data versus synthetic data (both marginal-based and model-based), the precision varies significantly. Specifically, the overall precision on real data is 0.935. In contrast, the precision drops to 0.795 for marginal-based synthetics and improves to 0.895 for model-based synthetics.

- **Dual Behavior of Model-based Synthetics**: The precision of the attack using model-based synthetic data exhibits dual behavior. For most classes, its precision is high and comparable to attacks using real data for shadow training. However, for a few classes, the precision drops significantly (less than 0.1).

- **Reason for Low Precision**: The low precision on some classes using model-based synthetics is attributed to the target classifier's inability to confidently model the distribution of data records belonging to these classes. These classes are underrepresented in the target model's training dataset, with some contributing less than 0.6% of the dataset and having fewer than 30 training records out of 10,000.

- **Implication**: The findings suggest that while model-based synthesis can generate effective shadow training data for most classes, it struggles with classes that are underrepresented in the original training data. This limitation affects the precision of membership inference attacks targeted at these classes.

  In summary, the effectiveness of using model-based synthesis for shadow training data depends heavily on the representation of different classes within the target model's training dataset, influencing the precision of subsequent attacks.

3.4.2 Effect of the Number of Classes and Training Data per Class

- The number of output classes of the target model contributes to how much the model leaks. The more classes, the more signals about the internal state of the model are available to the attacker.

- In the article, for example, we explore why the results in Figure 4 are better for CIFAR-100 compared to CIFAR-10. The CIFAR-100 model is also more overfitted to its training dataset.

Given the same number of training records per class, the attack performs better against CIFAR-100 than CIFAR-10. For instance, we compare CIFAR-10 with a training dataset size of 2,000 to CIFAR-100 with a training dataset size of 20,000. In both cases, the average number of data records per class is 200, but the attack accuracy is significantly higher for CIFAR-100, approaching nearly 1.

- The paper investigates how the number of classes impacts attack accuracy. Target models were trained on the purchase dataset using Google Prediction API with varying class counts: 2, 10, 20, 50, and 100. According to Figure 10, attack precision varies with model complexity. Models with fewer classes reveal less training input information. As the number of classes increases, models must distinguish data more precisely for accurate classification. Informally, models with more output classes need to remember more about their training data, thus they leak more information.

3.4.3 Effect of Overfitting

- The more overfitted a model, the more it leaksâbut only for models of the same type.
- Therefore, overfitting is not the only factor that causes a model to be vulnerable to membership inference. The structure and type of the model also contribute to the problem.
- Among the models mentioned in the article, the Amazon-trained (100; 1×10^{-4}) model, which is more overfitted according to Table I in the paper, leaks more data than the Amazon-trained (10; 1×10^{-6}) model. However, both Amazon models leak less data than the Google-trained model. Despite this, the Google model, though less overfitted than one of the Amazon models, demonstrates significantly better predictive power and generalizability than both Amazon models.