In the name of God

Introduction to Machine Learning (25737-2)

Project Phase 0

Spring Semester 1402-03

Department of Electrical Engineering Sharif University of Technology

Instructor: Dr. R. Amir

Sara Rezanezhad

Kimia Fakheri

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

1. **Averaging**:
    - **Strengths**: Simple to implement and understand. It ensures that the final model reflects an average performance across all shards.
    - **Weaknesses**: It might dilute the impact of data points that are more significant in some shards, potentially leading to a less accurate final model if the distribution of data across shards is not uniform.
2. **Weighted Averaging**:
    - **Strengths**: By assigning weights to different shards based on their importance or the quality of the data they contain, this method can produce a more accurate and representative final model.
    - **Weaknesses**: Determining the appropriate weights can be challenging and may require additional computation and domain knowledge. If weights are not chosen correctly, the method can introduce bias.
3. **Voting (Majority Voting or Weighted Voting)**:
    - **Strengths**: This method is robust to outliers and can be more accurate in cases where the correct classification is clear-cut. Weighted voting can further refine this by giving more influence to more reliable shards.
    - **Weaknesses**: In continuous output spaces (e.g., regression tasks), voting might not be directly applicable or might require discretization, potentially losing some granularity of the data. Additionally, voting schemes might not perform well if there is a high degree of variability or conflict between the outputs of different shards.

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

The SISA algorithm, while efficient in many scenarios, can become inefficient under certain circumstances:

1. **Large Number of Shards and Slices**:
    - **Inefficiency**: As the number of shards and slices increases, the computational overhead for managing and updating each shard and slice becomes significant. This can lead to increased memory usage and longer processing times, especially during the aggregation and unlearning phases.
    - **Mitigation**: Optimize the number of shards and slices based on the dataset size and complexity. Use hierarchical sharding to group similar shards together, reducing the number of individual elements that need to be managed. Implement parallel processing techniques to handle multiple shards and slices simultaneously.
2. **Highly Interdependent Data**:

- **Inefficiency**: If the data within shards are highly interdependent, isolating them may lead to suboptimal models due to the loss of important cross-shard relationships. This can degrade the overall performance of the aggregated model.
- **Mitigation**: Use advanced techniques like transfer learning or federated learning to maintain some level of dependency between shards. Additionally, periodically perform global training rounds where the entire dataset is used to refine the model, ensuring that interdependencies are captured.

3. **Frequent Unlearning Requests**:
   - **Inefficiency**: In environments with frequent unlearning requests, the constant need to retrain specific shards and slices can become computationally expensive and time-consuming.
   - **Mitigation**: Implement incremental learning techniques that allow for quick updates to the model without full retraining. Use caching mechanisms to store intermediate models and parameter states, reducing the time required for retraining. Additionally, employ differential privacy techniques to minimize the need for unlearning by ensuring that individual data points have a limited impact on the overall model.
4. **Non-Uniform Data Distribution**:
   - **Inefficiency**: If the data is not uniformly distributed across shards, some shards may become more significant than others, leading to imbalances and potential biases in the final aggregated model.
   - **Mitigation**: Implement stratified sharding to ensure that each shard represents a balanced subset of the overall data distribution. Regularly monitor and rebalance shards as needed to maintain uniformity. Use techniques like data augmentation to address imbalances within shards.
5. **Complex Aggregation Methods**:
   - **Inefficiency**: Complex aggregation methods can introduce additional computational overhead and latency, especially when dealing with large models or high-dimensional data.
   - **Mitigation**: Simplify aggregation methods where possible, or use approximate aggregation techniques that provide a good balance between accuracy and efficiency. For instance, using a combination of averaging and voting can simplify the process while maintaining reasonable accuracy.

By addressing these inefficiencies with targeted strategies, the SISA algorithm can be made more practical and scalable for a wider range of applications, ensuring its effectiveness in managing data privacy and model performance.

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

Evaluating the performance of an unlearning algorithm involves assessing how well the algorithm forgets the specified data while maintaining the overall performance and integrity of the model. Here are several metrics that are critical for this evaluation:

1. **Unlearning Efficacy**:
    - **Definition**: Measures how effectively the specified data has been removed from the model's knowledge.
    - **Reasoning**: The primary goal of unlearning is to ensure that the model no longer retains any influence from the forgotten data. This can be measured using metrics like influence functions or model inversion attacks to determine if the removed data still affects the model's predictions.
2. **Model Accuracy/Performance**:
    - **Definition**: Evaluates the overall performance of the model on a validation or test dataset before and after unlearning.
    - **Reasoning**: It's essential to ensure that the model's performance on non-forgotten data remains as stable as possible. Metrics like accuracy, precision, recall, and F1-score (for classification tasks) or mean squared error (for regression tasks) can be used to assess this.
3. **Computational Efficiency**:
    - **Definition**: Measures the computational resources required for the unlearning process, including time and memory usage.
    - **Reasoning**: Efficient unlearning should not significantly increase computational costs. This can be measured by tracking the time taken to unlearn specified data points and the additional memory overhead incurred during the process.
4. **Data Privacy Metrics**:
    - **Definition**: Assesses the degree of privacy protection achieved after unlearning. One such metric is the Differential Privacy guarantee, which quantifies the level of privacy by measuring the indistinguishability of the model's output with and without the specified data.
    - **Reasoning**: Ensuring that unlearning enhances privacy is crucial. Metrics like epsilon (ϵ\epsilonϵ) in differential privacy can indicate how much influence the removed data had on the model.
5. **Model Stability**:
    - **Definition**: Measures how stable the model's parameters are before and after unlearning. This can include metrics like parameter distance or Frobenius norm of the difference between model weights pre- and post-unlearning.
    - **Reasoning**: Stability is important to ensure that unlearning does not cause significant drifts or inconsistencies in the model's behavior.

Chosen Metric: Unlearning Efficacy

While all the above metrics are important, **Unlearning Efficacy** is particularly crucial for directly evaluating the primary objective of the unlearning algorithm. Here's why:

- **Direct Measure of Goal Achievement**: Unlearning efficacy directly assesses whether the unlearning algorithm has successfully removed the influence of the specified data from the model. This is the fundamental goal of any unlearning process.
- **Compliance with Privacy Regulations**: Ensuring that the model forgets specified data points aligns with regulatory requirements (e.g., GDPR's right to be forgotten). Thus, a metric that directly measures this compliance is essential.

- **Focus on Data Privacy**: By concentrating on unlearning efficacy, we can ensure that the algorithm prioritizes privacy and effectively mitigates the risk of data re-identification or leakage.

How to Measure Unlearning Efficacy

- **Influence Functions**: Calculate the influence of the removed data points on the model's predictions to ensure they no longer have any significant impact.
- **Model Inversion Attacks**: Attempt to reconstruct the forgotten data from the model's parameters and ensure that such reconstruction is no longer feasible after unlearning.
- **Performance on Forgotten Data**: Evaluate the model's performance on the data points that were requested to be forgotten. A significant drop in performance on these points would indicate successful unlearning.

By focusing on unlearning efficacy while also considering model accuracy, computational efficiency, privacy metrics, and model stability, we can comprehensively evaluate the performance of the unlearning algorithm.

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

The architecture of a machine learning model significantly influences its learning and unlearning time, as well as its performance. Different architectures come with unique characteristics that affect these aspects in various ways. Let's explore these effects in detail:

1. **Learning Time**:
    - **Complexity and Depth**: Deeper and more complex architectures (e.g., deep neural networks, transformers) typically have more parameters and require more computational resources and time to train. Shallow models or simpler architectures (e.g., linear models, decision trees) tend to have shorter learning times due to fewer parameters.
    - **Parallelism and Hardware Utilization**: Architectures that can efficiently utilize parallel processing capabilities (e.g., convolutional neural networks) can reduce learning time by leveraging GPUs or TPUs. Architectures that are less amenable to parallelism may experience longer learning times.
2. **Unlearning Time**:
    - **Parameter Redundancy**: Models with high parameter redundancy (e.g., overparameterized neural networks) might take longer to unlearn since removing the influence of specific data points requires updating a larger number of parameters.
    - **Modularity**: Architectures that allow for modular training (e.g., ensemble methods, SISA) can have more efficient unlearning processes. These architectures enable targeted retraining of specific modules or components, reducing the time required to unlearn specific data points.

3. **Learning Performance**:
    - **Expressive Power**: More complex architectures generally have higher expressive power, allowing them to capture intricate patterns in data, which can lead to better learning performance. For example, deep learning models can achieve state-of-the-art results in image and language processing tasks.
    - **Overfitting**: Complex models are also more prone to overfitting, especially if not properly regularized. This can lead to poorer generalization performance on unseen data. Simpler models, while potentially less powerful, can generalize better if the data is not overly complex.
4. **Unlearning Performance**:
    - **Adaptability**: Models that can easily adapt to changes in data distribution (e.g., through online learning or incremental learning techniques) tend to perform better in unlearning scenarios. They can efficiently remove the influence of specific data points without significant degradation in overall performance.
    - **Robustness**: Architectures designed with robustness in mind (e.g., using dropout, weight pruning) may handle unlearning more gracefully, maintaining performance stability after data removal. Conversely, highly tuned models may experience more significant performance drops when data points are unlearned.

Detailed Analysis of Common Model Architectures:

1. **Linear Models (e.g., Linear Regression, Logistic Regression)**:
    - **Learning Time**: Fast learning time due to simplicity and fewer parameters.
    - **Unlearning Time**: Relatively quick, as the influence of specific data points can be efficiently adjusted by updating a small number of parameters.
    - **Learning Performance**: Good for linearly separable data but limited expressive power for complex patterns.
    - **Unlearning Performance**: Stable, as changes to parameters are straightforward and limited in scope.
2. **Decision Trees and Ensembles (e.g., Random Forests, Gradient Boosting)**:
    - **Learning Time**: Moderate, depending on the depth of the trees and the number of trees in the ensemble.
    - **Unlearning Time**: Can be efficient if individual trees or nodes can be retrained or pruned without affecting the entire model.
    - **Learning Performance**: High performance on a variety of tasks due to the ability to model complex decision boundaries.
    - **Unlearning Performance**: May require careful handling to ensure that unlearning specific data points does not lead to overfitting or underfitting in certain parts of the data space.
3. **Convolutional Neural Networks (CNNs)**:
    - **Learning Time**: Can be long due to the need to train many filters and layers, but highly parallelizable.
    - **Unlearning Time**: Longer due to the deep architecture and numerous parameters, but modular retraining of specific layers or filters can mitigate this.
    - **Learning Performance**: Excellent for spatial data (e.g., images), capturing hierarchical features effectively.

- **Unlearning Performance**: May be challenging if data to be unlearned affects lower-level features, but higher-level feature retraining can be more manageable.

4. **Recurrent Neural Networks (RNNs) and Variants (e.g., LSTM, GRU)**:
   - **Learning Time**: Can be long due to sequential nature and dependencies across time steps, though parallelization techniques (e.g., truncated backpropagation) can help.
   - **Unlearning Time**: Typically long due to the need to retrain sequential dependencies and complex temporal relationships.
   - **Learning Performance**: Strong performance on sequential and time-series data, capable of capturing temporal patterns.
   - **Unlearning Performance**: Difficult, as removing specific data points may disrupt learned temporal dependencies significantly.
5. **Transformers**:
   - **Learning Time**: Long due to attention mechanisms and numerous parameters, but benefits greatly from parallel processing capabilities.
   - **Unlearning Time**: Similar to CNNs, unlearning can be time-consuming due to deep architecture, but parallel retraining of attention heads and layers can help.
   - **Learning Performance**: State-of-the-art performance on tasks requiring understanding of complex dependencies (e.g., NLP tasks).
   - **Unlearning Performance**: Moderate to challenging, as attention mechanisms need to be recalibrated carefully to remove the influence of specific data points without disrupting overall model performance.

Conclusion

The architecture of a model plays a crucial role in determining the efficiency and effectiveness of both learning and unlearning processes. Simple models offer faster learning and unlearning times but may lack the expressive power needed for complex tasks. In contrast, complex models provide high performance but come with increased computational costs and potential difficulties in unlearning specific data points. To mitigate these issues, leveraging modular and parallelizable architectures, as well as employing incremental learning techniques, can help balance the trade-offs between learning time, unlearning time, and overall performance.

Theory Question 5: Explain Differentially Private Algorithms and Their Techniques for Training a Differentially Private Model

Differential Privacy (DP) is a rigorous mathematical framework that provides quantifiable privacy guarantees when analyzing and sharing data. In the context of machine learning, differentially private algorithms ensure that the inclusion or exclusion of any single data point in the training set has a minimal impact on the model's output, thereby protecting the privacy of individual data points.

Here's a detailed explanation of differentially private algorithms and their techniques for training a differentially private model:

1. Differential Privacy Basics

Definition: A randomized algorithm A\mathcal{A}A is said to be (ϵ,δ)(\epsilon, \delta)(ϵ,δ)-differentially private if for all datasets DDD and D′D'D′ differing in at most one element, and for all possible outputs SSS of the algorithm:

P(A(D)∈S)≤eϵ·P(A(D′)∈S)+δP(\mathcal{A}(D) \in S) \leq e^{\epsilon} \cdot P(\mathcal{A}(D') \in S) + \deltaP(A(D)∈S)≤eϵ·P(A(D′)∈S)+δ

- **ϵ\epsilonϵ** (epsilon): Privacy loss parameter; lower values mean stronger privacy guarantees.
- **δ\deltaδ** (delta): Probability of the privacy guarantee being broken; smaller values indicate stronger guarantees.

2. Techniques for Training Differentially Private Models

A. Differentially Private Stochastic Gradient Descent (DP-SGD): DP-SGD is an adaptation of the standard stochastic gradient descent algorithm to provide differential privacy.

- **Gradient Clipping**: Before updating model parameters, the gradient of the loss function with respect to the model parameters is computed for each data point. These gradients are then clipped to a fixed norm CCC. This limits the influence of any single data point.

  clip(∇Li,C)=∇Li·min⁡(1,C||∇Li||2)\text{clip}(\nabla L_i, C) = \nabla L_i \cdot \min\left(1, \frac{C}{||\nabla L_i||_2}\right)clip(∇Li,C)=∇Li·min(1,||∇Li||2C)

- **Adding Noise**: After clipping, random noise from a Gaussian distribution (with mean 0 and standard deviation proportional to CCC) is added to the aggregated gradients to ensure privacy.

  ∇L=1N∑i=1Nclip(∇Li,C)+N(0,σ2C2I)\nabla L = \frac{1}{N} \sum_{i=1}^{N} \text{clip}(\nabla L_i, C) + \mathcal{N}(0, \sigma^2 C^2 I)∇L=N1∑i=1Nclip(∇Li ,C)+N(0,σ2C2I)

- **Model Update**: The model parameters are updated using the noisy gradients.

  θt+1=θt−η·∇L\theta_{t+1} = \theta_t - \eta \cdot \nabla Lθt+1=θt−η·∇L

B. Privacy Accounting: To keep track of the overall privacy loss across multiple training iterations, the privacy budget (ϵ,δ\epsilon, \deltaϵ,δ) is carefully managed using techniques like the Moments Accountant or the Gaussian Differential Privacy (GDP) framework.

C. Noise Addition to Data (Input Perturbation): Noise can also be added directly to the input data before training. This technique ensures that the data itself is differentially private.

- **Laplace Mechanism**: Adds noise drawn from a Laplace distribution to the data.

$x' = x + \text{Lap}(b)x′=x+Lap(b)$

- **Gaussian Mechanism**: Adds noise drawn from a Gaussian distribution to the data.

  $x' = x + \mathcal{N}(0, \sigma^2)x′=x+N(0,σ2)$

D. Output Perturbation: Noise can be added to the output of the learning algorithm. For example, after training a model, noise can be added to the model parameters or the predictions made by the model.

E. Objective Perturbation: This technique involves adding noise to the objective function used during training, thereby indirectly ensuring that the learned model parameters incorporate privacy protection.

Benefits and Trade-offs

- **Benefits**:
  - **Privacy Guarantees**: Differential privacy provides strong, mathematically-proven privacy guarantees.
  - **Robustness**: Models trained with differential privacy are often more robust to overfitting, as the noise introduced can act as a regularizer.
- **Trade-offs**:
  - **Accuracy**: The introduction of noise can degrade model accuracy, particularly if the privacy parameters (ϵ and δ) are very strict.
  - **Computational Overhead**: Techniques like DP-SGD can introduce additional computational overhead due to gradient clipping and noise addition.

Practical Considerations

When implementing differentially private training models, it's important to carefully balance the trade-off between privacy and utility. Selecting appropriate privacy parameters (ϵ, δ), smaller values provide stronger privacy but can significantly impact model performance. Techniques such as hyperparameter tuning and validation on a separate dataset can help find the optimal balance for a given application.

By employing differentially private algorithms and techniques, organizations can train machine learning models that respect individual privacy while still extracting useful insights from the data.

Theory Question 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 techniques are fundamental methods used in machine learning to improve model generalization, prevent overfitting, and enhance privacy. While these techniques share some similarities with the noise addition methods used in differential privacy, they serve different primary purposes. Let's explore these techniques and compare them with differential privacy methods.

Regularization Techniques

1. L1 Regularization (Lasso):

- **Mechanism**: Adds a penalty equal to the absolute value of the magnitude of coefficients.
- **Objective Function**:

  L(θ)=L0(θ)+λ∑i|θi|L(\theta) = L_0(\theta) + \lambda \sum_{i} |\theta_i|L(θ)=L0(θ)+λ∑i|θi|

  where L0(θ)L_0(\theta)L0(θ) is the original loss function and λ\lambdaλ is the regularization parameter.

- **Effect**: Encourages sparsity in the model parameters, effectively performing feature selection by driving some coefficients to zero.
- **Privacy Aspect**: By simplifying the model, L1 regularization can reduce the risk of overfitting to individual data points, which indirectly helps in enhancing privacy.

2. L2 Regularization (Ridge):

- **Mechanism**: Adds a penalty equal to the square of the magnitude of coefficients.
- **Objective Function**:

  L(θ)=L0(θ)+λ∑iθi2L(\theta) = L_0(\theta) + \lambda \sum_{i} \theta_i^2L(θ)=L0(θ)+λ∑iθi2

- **Effect**: Penalizes large coefficients, leading to a model where the influence of any single data point is minimized.
- **Privacy Aspect**: Helps in reducing the sensitivity of the model to individual data points, contributing to robustness against membership inference attacks.

3. Elastic Net Regularization:

- **Mechanism**: Combines L1 and L2 regularization.
- **Objective Function**:

  L(θ)=L0(θ)+λ1∑i|θi|+λ2∑iθi2L(\theta) = L_0(\theta) + \lambda_1 \sum_{i} |\theta_i| + \lambda_2 \sum_{i} \theta_i^2L(θ)=L0(θ)+λ1∑i|θi|+λ2∑iθi2

- **Effect**: Balances the sparsity of L1 and the stability of L2 regularization.

- **Privacy Aspect**: Provides a balance between feature selection and parameter shrinkage, indirectly aiding in privacy protection.

Normalization Techniques

1. Batch Normalization:

- **Mechanism**: Normalizes the inputs of each layer to have a mean of zero and a variance of one within each mini-batch.
- **Formula**:

  x^=x−μσ2+ϵ\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}x^=σ2+ϵx−μ

  where μ\muμ and σ\sigmaσ are the mean and variance of the mini-batch, and ϵ\epsilonϵ is a small constant.

- **Effect**: Stabilizes and accelerates training by reducing internal covariate shift.
- **Privacy Aspect**: Can help in reducing the model's sensitivity to individual data points by normalizing their contributions.

2. Layer Normalization:

- **Mechanism**: Normalizes the inputs across the features rather than the batch.
- **Formula**:

  x^=x−μσ2+ϵ\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}x^=σ2+ϵx−μ

  where μ\muμ and σ\sigmaσ are the mean and variance across the features for a single training example.

- **Effect**: Improves training stability, especially for recurrent neural networks.
- **Privacy Aspect**: By normalizing contributions across features, it can help in mitigating the impact of any single data point.

3. Group Normalization:

- **Mechanism**: Divides the channels into groups and normalizes within each group.
- **Formula**:

  x^=x−μσ2+ϵ\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}x^=σ2+ϵx−μ

  where μ\muμ and σ\sigmaσ are the mean and variance within the group.

- **Effect**: Balances the benefits of batch and layer normalization.
- **Privacy Aspect**: Similar to batch and layer normalization, it helps in reducing model sensitivity to individual data points.

Comparison with Differential Privacy

Similarities:

- **Noise Addition**: Both differential privacy and regularization techniques add a form of noise or perturbation to the training process. In differential privacy, noise is added explicitly to gradients, data, or outputs to mask the contribution of individual data points. In regularization, the "noise" is introduced implicitly by penalizing large coefficients, thereby reducing the model's capacity to memorize individual data points.
- **Privacy Enhancement**: Both approaches aim to enhance the privacy of the training process. Regularization indirectly protects privacy by preventing overfitting and reducing model complexity, while differential privacy provides explicit, quantifiable privacy guarantees.

Differences:

- **Purpose**: The primary purpose of regularization and normalization is to improve model generalization and stability, whereas differential privacy explicitly aims to protect the privacy of individual data records.
- **Mechanism**: Differential privacy involves adding random noise to data, gradients, or outputs, ensuring that the presence or absence of a single data point does not significantly affect the model. Regularization, on the other hand, modifies the loss function to penalize large parameter values, indirectly reducing the model's sensitivity to individual data points.
- **Privacy Guarantees**: Differential privacy provides formal mathematical guarantees about the level of privacy protection, quantified by parameters ϵ\epsilonϵ and δ\deltaδ. Regularization does not offer such formal guarantees but can still contribute to privacy by limiting overfitting and model complexity.

Conclusion

Regularization and normalization techniques play crucial roles in training robust and stable machine learning models. While they share some similarities with differential privacy methods in terms of adding perturbations to the training process, their primary purposes and mechanisms differ. Regularization and normalization focus on improving model generalization and stability, whereas differential privacy provides explicit and formal privacy guarantees. Together, these techniques can be combined to enhance both the performance and privacy of machine learning models.

Theory Question 7: Other Techniques for Training a Private Model

Beyond differential privacy, regularization, and normalization, several other techniques can be employed to train private models. These methods aim to safeguard sensitive data and protect against various privacy attacks while maintaining the utility of the trained models. Here are some notable techniques:

1. Homomorphic Encryption

Homomorphic encryption allows computations to be performed on encrypted data without decrypting it. This means that a model can be trained on encrypted data, and the results can be decrypted to obtain the final model without ever exposing the raw data.

- **Fully Homomorphic Encryption (FHE)**: Supports arbitrary computations on encrypted data but is computationally expensive.
- **Partially Homomorphic Encryption (PHE)**: Supports specific types of operations (e.g., addition or multiplication) and is more efficient than FHE.

Advantages:

- Strong privacy guarantees as data remains encrypted throughout the computation process.
- Protects against unauthorized access to sensitive data.

Disadvantages:

- Computational overhead is significant, especially for fully homomorphic encryption.
- Implementation complexity is high.

2. Secure Multi-Party Computation (SMPC)

Secure multi-party computation allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. Each party only learns the output of the computation and nothing else about the other parties' inputs.

- **Example Protocols**: Yao's Garbled Circuits, Secret Sharing.

Advantages:

- Ensures data privacy in collaborative learning scenarios.
- No single party has access to the entire dataset, reducing the risk of data breaches.

Disadvantages:

- Communication overhead can be high, depending on the number of parties and the complexity of the computation.
- Requires coordination and trust among the parties involved.

3. Federated Learning

Federated learning involves training machine learning models across decentralized devices or servers holding local data samples, without exchanging the data itself. Only model updates (e.g., gradients) are shared and aggregated.

Process:

- Each client device trains a local model on its data.
- Local models' updates are sent to a central server, which aggregates them to update the global model.
- The updated global model is sent back to the client devices.

Advantages:

- Data remains local, reducing the risk of data breaches.
- Scalability, as multiple devices can contribute to the model training.

Disadvantages:

- Communication costs can be high due to frequent exchanges of model updates.
- Aggregated updates can still leak information about the local data, requiring additional privacy techniques like differential privacy.

4. Knowledge Distillation

Knowledge distillation involves training a smaller, student model to mimic the behavior of a larger, teacher model. The student model learns from the soft predictions of the teacher model, which can obscure the contribution of individual data points.

Process:

- Train a teacher model on the original dataset.
- Use the teacher model to generate soft predictions for a student model.
- Train the student model using these soft predictions.

Advantages:

- Reduces model complexity and size.
- Indirectly obscures the influence of individual training examples.

Disadvantages:

- The privacy guarantee is indirect and less formal than differential privacy.
- The student model's performance depends on the quality of the teacher model.

5. Synthetic Data Generation

Synthetic data generation involves creating artificial data that has similar statistical properties to the original data but does not contain any real individual's information.

- **Techniques**: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs).

Advantages:

- No real data is exposed, thus protecting privacy.
- Synthetic data can be used for training without privacy concerns.

Disadvantages:

- The quality of synthetic data is crucial; poor-quality synthetic data can degrade model performance.
- There is a risk of the synthetic data still revealing some properties of the real data if not done carefully.

6. Data Anonymization and Pseudonymization

Data anonymization and **pseudonymization** involve transforming the data to remove or obfuscate personal identifiers.

- **Anonymization**: Irreversibly removing identifiers.
- **Pseudonymization**: Replacing identifiers with pseudonyms while keeping the mapping between identifiers and pseudonyms in a separate secure location.

Advantages:

- Reduces the risk of re-identification of individuals.
- Often required by data protection regulations (e.g., GDPR).

Disadvantages:

- Anonymization can be challenging to achieve effectively, as there is a risk of re-identification through linkage attacks.
- Pseudonymization still carries some risk if the mapping is compromised.

Conclusion

There are several techniques available for training private models beyond differential privacy, regularization, and normalization. Each technique offers unique advantages and trade-offs in terms of privacy protection, computational efficiency, and ease of implementation. The choice of technique depends on the specific requirements and constraints of the application, such as the sensitivity of the data, the computational resources available, and the desired level of privacy protection. Combining multiple techniques can often provide stronger privacy guarantees while maintaining model performance.

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

1. **Adversarial Shadow Attenuation**:

- **Method**: A novel GAN-based framework trains a shadow detection network (D-Net) alongside a shadow attenuation network (A-Net). The A-Net modifies original training images using a simplified physical shadow model, generating adversarial examples. These hard-to-predict cases augment the D-Net's training data.
- **Effect**: The additional data from A-Net significantly improves shadow detection accuracy [1].

2. **Pixel Height Maps**:
   - **Method**: Introduces "Pixel Height," a geometry representation encoding correlations between objects, ground, and camera pose. It can be calculated from 3D geometries, manually annotated on 2D images, or predicted from RGB images. Pixel Height enables precise control of shadow direction and shape.
   - **Application**: Provides controllable shadow generation, enhancing realism in image compositing [5].
3. **Data Synthesis via GANs**:
   - **Method**: Generative adversarial networks (GANs) learn to generate shadows by training with pairs of shadow and shadow-free images.
   - **Focus**: These methods mainly generate hard shadows, but they lack editability [7].

Source:
(1) A+D Net: Training a Shadow Detector with Adversarial Shadow Attenuation. https://www3.cs.stonybrook.edu/~minhhoai/papers/shadowAttECCV18.pdf.
(2) arXiv:2207.05385v2 [cs.CV] 15 Jul 2022. https://arxiv.org/pdf/2207.05385.
(3) Controllable Shadow Generation Using Pixel Height Maps - Purdue University. https://www.cs.purdue.edu/homes/bbenes/papers/Sheng22ECCV.pdf.
(4) Membership Inference Attacks Against Machine Learning Models - arXiv.org. https://arxiv.org/pdf/1610.05820.
(5) Generative Models: Training Data to the Image Generation Process. https://3daily.ai/blog/understanding-generative-models-from-training-data-to-image-generation-process/.
(6) Inference Attacks against Machine Learning models. https://techairesearch.com/inference-attacks-against-machine-learning-models/.
(7) Interpretability of Blackbox Machine Learning Models through ... - DeepAI. https://deepai.org/publication/interpretability-of-blackbox-machine-learning-models-through-dataview-extraction-and-shadow-model-creation.

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

When generating synthetic data for shadow models, the algorithm typically involves the following steps:

1. **Model Selection**:

- Choose a suitable generative model. Common choices include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), or autoencoders.
- Train the chosen model on a dataset containing shadow-free images (e.g., images without shadows).

2. **Latent Space Sampling**:
   - In the trained generative model, each image corresponds to a point in a latent space.
   - Randomly sample points from this latent space. These samples represent synthetic data points.
3. **Decoding to Images**:
   - Decode the sampled latent vectors back into image space using the generative model.
   - The decoded images represent synthetic shadowed versions of the original images.
4. **Shadow Generation**:
   - Introduce shadows to the synthetic images. This can be done by:
     - Overlaying shadow patterns (e.g., Gaussian or soft shadows) on the synthetic images.
     - Adjusting pixel intensities to simulate shadow effects.
     - Modifying lighting conditions or object positions to create realistic shadows.
5. **Data Augmentation**:
   - Apply additional transformations (e.g., rotation, scaling, or translation) to the synthetic images to increase diversity.
   - Augmented data helps improve the robustness of the shadow model.
6. **Labeling and Pairing**:
   - Label the synthetic images as "shadow" or "shadow-free."
   - Pair each synthetic shadow image with its corresponding shadow-free counterpart from the original dataset.
7. **Training Shadow Models**:
   - Use the labeled synthetic data (shadowed images) along with the original shadow-free data to train shadow detection models.
   - Common architectures include CNNs or other deep learning models.
8. **Evaluation and Fine-Tuning**:
   - Evaluate the trained shadow model's performance on a separate validation set.
   - Fine-tune the model if necessary based on evaluation results.

Source:
(1) Regularization in Machine Learning (with Code Examples) - Dataquest. https://www.dataquest.io/blog/regularization-in-machine-learning/.
(2) Understanding Regularization and Normalization in Machine Learning Models. https://www.toolify.ai/ai-news/understanding-regularization-and-normalization-in-machine-learning-models-778615.
(3) What is Normalization in Machine Learning? A Comprehensive ... - DataCamp. https://www.datacamp.com/tutorial/normalization-in-machine-learning.
(4) Normalization Techniques for Deep Learning Regression Models. https://machinelearningmodels.org/normalization-techniques-for-deep-learning-regression-models/.
(5) Special Issue: Regularization Techniques for Machine Learning and Their .... https://mdpi-res.com/d_attachment/electronics/electronics-11-00521/article_deploy/electronics-11-00521-v2.pdf?version=1644542121.

Theory Question 10. Explain how attack model is trained using shadow models.:

When training an attack model using shadow models, the process involves creating shadow models that mimic the behavior of the target model. These shadow models are then used to generate training data for the attack model. Here's the steps:

1. **Shadow Models**:
    - Shadow models are auxiliary models trained to approximate the classification output of the target model.
    - These models learn from the same data distribution as the target model but are not identical to it.
2. **Data Synthesis**:
    - The shadow models are used to generate synthetic data.
    - For each input, the shadow models produce a classification output vector (similar to the target model's predictions).
3. **Membership Inference Attack (MIA)**:
    - The synthetic data generated by shadow models is split into two groups:
        - **Members**: Data points that were part of the target model's training dataset.
        - **Non-members**: Data points not used during target model training.
    - The goal is to determine whether a given data record was part of the target model's training dataset (membership inference).
4. **Training the Attack Model**:
    - An attack model (often a neural network or other classifier) is trained using the synthetic data.

- The attack model learns to distinguish between members and non-members based on the shadow model's outputs.
- It identifies patterns that reveal whether an input was part of the target model's training data.

5. **Evaluation**:
   - The trained attack model is evaluated on a separate validation set.
   - High accuracy in distinguishing members from non-members indicates vulnerability of the target model to membership inference attacks.
6. **Mitigation Strategies**:
   - Researchers study factors influencing leakage (e.g., model architecture, dataset size).
   - Mitigation techniques include adversarial training, privacy-preserving mechanisms, and model modifications.

Source:
(1) Membership Inference Attacks Against Machine Learning Models - arXiv.org. https://arxiv.org/pdf/1610.05820.
(2) Mphasis-ML-Marketplace/ML-Robustness-MIA-Using-Shadow-Models. https://github.com/Mphasis-ML-Marketplace/ML-Robustness-MIA-Using-Shadow-Models.
(3) Membership Inference Attack Using Self Influence Functions. https://openaccess.thecvf.com/content/WACV2024/papers/Cohen_Membership_Inference_Attack_Using_Self_Influence_Functions_WACV_2024_paper.pdf.
(4) TransMIA: Membership Inference Attacks Using Transfer Shadow Training. https://arxiv.org/pdf/2011.14661.

Theory Question 11. Explain the effect of following concepts in accuracy of the attack model: 1. Effect of the shadow training data generated using the three methods you explained earlier. 2. Effect of the number of classes and training data per class. 3. Effect of overfitting.

1. Effect of the Shadow Training Data:
   The shadow training data is crucial for simulating the target model's behavior and improving the attack model's accuracy. The three methods of generating shadow training data—using data from the same distribution as the target model, using synthetic data, or using data from a different but related distribution—can all impact the attack model's accuracy. For instance, using data from the same distribution as the target model can lead to higher accuracy in inferring membership, as the attack model can learn more representative patterns of the target model's behavior[1].

2. Effect of the Number of Classes and Training Data per Class:
   The number of classes and the amount of training data per class can significantly affect the attack model's accuracy. Generally, a higher number of classes can make it more challenging for the attack model to accurately infer membership, as the probability differences between classes decrease, making confident predictions harder[10]. Moreover, the amount of training data per

class is also important; insufficient data can lead to poor generalization, while a balanced and ample amount of data can improve the attack model's accuracy[1][2].

3. Effect of Overfitting:
   Overfitting occurs when a model learns the training data too well, including its noise and outliers, which reduces its ability to generalize to new data. In the context of MIAs, an overfitted target model may exhibit more distinct behavior on its training data compared to new data, making it easier for the attack model to distinguish between members and non-members. However, this also means that the attack model's accuracy might be artificially high due to the target model's poor generalization rather than the attack model's inherent effectiveness[8].

Source:

(1) Practical Membership Inference Attacks Against Large-Scale Multi-Modal ....
https://openaccess.thecvf.com/content/ICCV2023/papers/Ko_Practical_Membership_Inference_Attacks_Against_Large-Scale_Multi-Modal_Models_A_Pilot_ICCV_2023_paper.pdf.
(2) Does the number of classes in the target variable affect the accuracy ....
https://stackoverflow.com/questions/71632842/does-the-number-of-classes-in-the-target-variable-affect-the-accuracy-of-a-class.
(3) Effects of Training Data Size and Class Imbalance on the ... - Springer.
https://link.springer.com/chapter/10.1007/978-3-030-34518-1_1.
(4) Machine Learning Overfitting - Try Machine Learning. https://trymachinelearning.com/machine-learning-overfitting/.
(5) Differential Privacy Under Membership Inference Attacks.
https://link.springer.com/chapter/10.1007/978-981-99-8296-7_18.
(6) Membership Inference Attacks Against Machine Learning Models - arXiv.org.
https://arxiv.org/pdf/1610.05820.
(7) ML Attack Models: Adversarial Attacks and Data Poisoning Attacks.
https://arxiv.org/pdf/2112.02797.
(8) To Overfit, or Not to Overfit: Improving the Performance of Deep ....
https://link.springer.com/chapter/10.1007/978-3-031-17433-9_17.
(9) Overconfidence is a Dangerous Thing: Mitigating Membership Inference ....
https://arxiv.org/pdf/2307.01610.
(10) Mastering Model Complexity: Avoiding Underfitting and Overfitting ....
https://www.pecan.ai/blog/machine-learning-model-underfitting-and-overfitting/.
(11) What is Overfitting? | DataCamp. https://www.datacamp.com/blog/what-is-overfitting.
(12) In classification, how does the number of classes affect the model size ....
https://ai.stackexchange.com/questions/26553/in-classification-how-does-the-number-of-classes-affect-the-model-size-and-amou.