TML Assignment 2 by Saif Ali and Usaid Bhirya

Advanced Membership Inference Attack on Neural Networks

In this assignment we are implementing a Membership Inference Attack (MIA) against a ResNet18 model trained on image classification. In this attack we use multiple feature extraction techniques and ensemble learning to determine whether specific data samples were part of the target model's training set. The complete implementation is available in the Jupyter Notebook.

Data accessible to the us (The adversary)

- **Target Model**: A pre-trained ResNet18 model (`01_MIA.pt`) with 44 output classes
- **Public Dataset**: A dataset (`pub.pt`) containing samples with known membership labels for training attack models
- **Private Dataset**: A test dataset (`priv_out.pt`) where membership needs to be inferred
- **Model Architecture**: Knowledge that the target uses ResNet18 with a modified final layer (512 → 44 classes)
- **Data Preprocessing**: Access to the normalization parameters (mean and std) used during training

Approach Used

Our implementation uses a multi-faceted membership inference strategy that combines gradient-based analysis for feature extraction, confidence scoring, activation pattern analysis of the trained model, and use of multiple shadow models techniques. The complete implementation is available in the `AdvancedMembershipInferenceAttack` class.

Core Intuition

The fundamental premise of membership inference attacks is that machine learning models exhibit different behaviors when processing training data versus unseen data. Training samples typically show: - Higher confidence predictions with sharper probability distributions - Lower loss values and more predictable gradient patterns - Distinct activation patterns in intermediate layers - Different responses to model perturbations

Our approach exploits these behavioral differences through comprehensive feature engineering with the use of gradients and shadow models, and ensemble learning.

Feature Extraction Strategy

1. Advanced Confidence-Based Features The attack extracts sophisticated confidence metrics that go beyond simple maximum probability:

- **Basic Confidence Metrics**: Maximum probability and logit values provide baseline confidence indicators
- **Entropy Analysis**: Shannon entropy of prediction probabilities reveals distribution sharpness
- **Temperature Scaling Effects**: Testing model predictions under different temperature values (0.5, 2.0, 5.0) reveals how prediction confidence changes with calibration
- **Logit Statistics**: Mean, standard deviation, and maximum-minus-mean of raw logits capture prediction uncertainty
- **Top-k Analysis**: Examining the top 5 predicted class probabilities provides insight into prediction distribution
- **True Class Analysis**: Analyzing the predicted probability, logit, and rank of the true class reveals model familiarity
- **Margin Analysis**: Computing gaps between top predictions (1st-2nd, 2nd-3rd) indicates prediction certainty
- **Loss-based Features**: Cross-entropy loss serves as a direct indicator of model familiarity

2. Gradient-Based Analysis Gradient information provides crucial insights into model behavior during backpropagation:

```
def compute_gradient_features(self, imgs, labels):
    # Compute gradients of loss w.r.t. input images
    grad_norm = torch.norm(imgs.grad[i]).item()
    grad_mean = torch.mean(torch.abs(imgs.grad[i])).item()
    grad_std = torch.std(imgs.grad[i]).item()
```

Training samples often exhibit different gradient patterns because: - The model has “seen” these patterns before during optimization - Gradient magnitudes and distributions differ between familiar and unfamiliar inputs - Overfitted models show distinct gradient signatures for training data

3. Intermediate Layer Activation Analysis We hooked into the ResNet18’s intermediate layers (layer1-layer4), and analyzed internal representations:

- **Activation Statistics**: Mean, standard deviation, maximum, and minimum values of layer activations
- **Global Average Pooling**: Spatial information is aggregated to create fixed-size feature vectors
- **Layer-wise Analysis**: Different network depths capture features at various abstraction levels

The intuition is that training samples activate neurons differently than test samples, particularly in deeper layers where the model has learned specific patterns.

Shadow Model Strategy

Shadow models serve as proxies to understand target model behavior, we wrote a function that allow us to train multple shadow models with slighty random parameters passed to them, this takes the number of shadow models to be trained as a parameter:

Diverse Training Strategies

- **Random Subsets**: Training on 40-70% of available data with random sampling
- **Class-Balanced Subsets**: Ensuring equal representation across classes
- **Varied Hyperparameters**: Different learning rates ($0.001 \times [0.5, 1.5]$) and training epochs (3-8)
- **Data Augmentation**: Random horizontal flips during training
- **Regularization**: Label smoothing (0.1) and weight decay (1e-4) for better generalization

Shadow Feature Extraction For computational efficiency, shadow models provide streamlined features: - Maximum prediction probability - Prediction entropy - Cross-entropy loss - True class probability

Ensemble Attack Model Architecture

We use multiple machine learning algorithms with different preprocessing strategies, such as the use of various Scalers, and later aggregate the results. The diverse set of models we used include:

Model Diversity

1. **Random Forest (500 estimators)**: Handles non-linear relationships and feature interactions
2. **Gradient Boosting (300 estimators)**: Sequential learning with controlled overfitting
3. **Support Vector Machine (RBF kernel)**: Non-linear decision boundaries with C=10.0
4. **Multi-Layer Perceptron (256-128-64 hidden units)**: Deep learning approach with dropout regularization

Preprocessing Strategies

- **StandardScaler**: Zero mean, unit variance normalization
- **RobustScaler**: Median-based scaling robust to outliers

Weighted Ensemble Prediction Different model types receive different weights based on empirical performance: - Tree-based models (RF, GB): 1.5× weight - SVM: 1.2× weight
- Neural networks: 1.0× weight

Final predictions undergo calibration sharpening: `score^0.8` to enhance discrimination.

Why This Multi-Faceted Approach?

1. **Robustness**: Multiple feature types capture different aspects of membership
2. **Complementarity**: Gradient, confidence, and activation features provide non-overlapping information
3. **Ensemble Benefits**: Different algorithms excel at different pattern types
4. **Shadow Model Validation**: Multiple models trained on known membership data validate attack feasibility

Implementation Optimizations

Computational Efficiency

- Batch processing (32-64 samples) for feature extraction
- Efficient gradient computation with selective backpropagation
- Fast shadow feature extraction using vectorized operations

Robustness Measures

- Multiple scaling strategies handle different feature distributions
- Class-balanced shadow training prevents bias
- Ensemble voting reduces individual model limitations

Technical Implementation Details

Data Preprocessing

Images are normalized using the target model's training statistics:

```
mean = [0.2980, 0.2962, 0.2987]
std = [0.2886, 0.2875, 0.2889]
```

Model Loading and Evaluation

The target ResNet18 model is loaded in evaluation mode to ensure consistent predictions across multiple queries.

Feature Dimensionality

The complete feature vector combines: - ~20 confidence-based features - 3 gradient-based features
- 16 activation-based features (4 layers × 4 statistics) - 24 shadow model features (6 models × 4 features)

Total: ~63 features per sample

Results and Performance

The attack achieves: - **AUC: ~0.67** (Area Under the ROC Curve) - **TPR@FPR=0.05: ~0.15** (True Positive Rate at 5% False Positive Rate)

Score Distribution Analysis

The final implementation provides detailed statistics: - Mean membership probability across private dataset - Standard deviation indicating prediction confidence - Score range showing discrimination capability

Submission Format

Results are formatted as a CSV file with sample IDs and corresponding membership scores, submitted via the provided API endpoint. Also uploaded to Github, with the name `test.csv`

Advanced Techniques Leveraged

1. Temperature Scaling Analysis

Testing predictions under different temperature parameters reveals how model confidence changes with calibration, providing additional discrimination power.

2. Multi-Scale Gradient Analysis

Computing gradient norms, means, and standard deviations captures different aspects of model sensitivity to input perturbations.

3. Layer-wise Activation Monitoring

Hooking intermediate layers provides insights into internal representation differences between training and test data.

4. Sophisticated Shadow Training

Using diverse training strategies and hyperparameters creates shadow models that better approximate target model behavior variations.

5. Weighted Ensemble Learning

Combining multiple attack models with performance-based weighting maximizes discrimination capability while maintaining robustness.

Files and Their Descriptions

- `Membership Inference Attack.ipynb` - Complete implementation including data loading, feature extraction, shadow model training, and ensemble attack execution
- `AdvancedMembershipInferenceAttack` class - Main attack implementation with modular design for different feature extraction methods
- `readme.md` - A brief description of the technique used, and the resultes we obtained.
- `test.csv` - The final submission file, a comma seperated file with sample IDs and corresponding membership scores.

Theoretical Foundation

This attack demonstrates the fundamental privacy vulnerabilities in machine learning models. Even without direct access to training data, sophisticated statistical analysis of model behavior can reveal sensitive membership information. The multi-faceted approach shows that combining different attack vectors significantly enhances inference capability, highlighting the need for robust privacy-preserving techniques in model deployment.

The success of this approach underscores the importance of differential privacy, model distillation, and other defense mechanisms in protecting against membership inference attacks in production machine learning systems.