Michael Shostrand

Intelligent Cybersecurity Assignment 2

This assignment explores membership inference attacks. The goal of a membership inference attack is to determine whether a data point was used to train the target model. The output of a model is different for inputs it has encountered in training. It will be much more certain that the input is a particular class. With input outside the training set, the probabilities are spread out among the classes more. It may still classify the input correctly, but with less certainty. Overfitting makes a model much more susceptible to this attack since the difference in output will be much more drastic. In this assignment two variations of the attack are carried out, white box and black box. A white box attack is where the model and architecture is available to the attacker. Black box is where the model architecture is unknown. The attacker can query the model and knows the data distribution. White box is easier to attack. However, black box is vulnerable as well.

Objectives: The goal of this assignment is to carry out membership inference attacks. There will be a white box and black box attack. To do this we will create a convolutional neural network to classify the CIFAR-10 dataset. This network has to be fairly accurate for the membership attack to work.

Approaches/Methods: To carry out a membership inference attack, an attack model is created to make inferences about data items. The input of the attack model is the output of the target model. Given the probability distribution that the target model outputs, the attack model will predict whether it was in the training set. The labels and/or the dataset of the target model will be unknown to the attacker. To generate the training data for the attack model, a shadow model is created. The shadow model has the same function as the target model. The dataset will be separated into two subsets. One set will be used to train the shadow model. The other “out” dataset will be left untouched until the shadow model is trained. Once the shadow model is trained, the attack model training data is created by running both the training and “out” dataset through the shadow model. The output of shadow model will be the features of the attack model. The entries will be labeled with a 0 or 1 for the “out” data set and “in” data set respectively. The attack model is then trained on the generated data. It can then make predictions using the output of the target model.

Target Model: The target model is a CNN trained with a subset of CIFAR-10. Test accuracy of 77% was achieved.

Shadow Model for White Box Attack: The same architecture as the target model is used. Test accuracy of 75% was achieved in shadow model training.

Shadow Model for Black Box Attack: Transfer learning was used for the black box. A pretrained resnet18 model from the torchvision module was used. The fully connected head was replaced with one appropriate for CIFAR-10 classification. The test accuracy was 73% after training for 20 epochs.

Attack Model: A fully connected neural network with two hidden layers was used for the attack model.

Workflow:

1. Train target model. In the real world, this step would not be necessary or possible.
2. Create shadow model to mirror target model
3. Generate training set for attack model
4. Train attack model on generated data
5. Evaluate precision and recall of attack model on target model

Datasets: CIFAR-10 for attack and shadow models. Output of shadow model is training data for attack model.

Parameters: Each of the models has the normal set of hyperparameters, learning rate, batch size, optimizer algorithm, epochs, etc. Additionally for the attacks the architecture of the shadow model is also a parameter that has an effect on the attack efficacy.

Discussion: I was not able to achieve good results for the attack model. The shadow model and target model had decent accuracy, ~75%. I was expecting this to be enough to carry out a membership inference. I could not get the attack model to converge. Modifying the learning rate had little to no effect. I tried modifying the training input to the attack model in a couple different ways. I tried passing both the logits and the class probability distribution to the attack model. That change didn’t seem to have much of an effect. Sorting the output in descending order did not seem to change the result. I haven't been able to determine why my attack model is converging so slowly. More debugging and tuning is required. I believe I understand the concepts of the attack. However the poor attack model performance doesn't not allow me to reinforce the concept with results.