Semester Project

Goal

Implement membership inference attacks against the given pre-trained target models.

Preparation

We use PyTorch framework for this project. You need to evaluate the attack performance on two target models trained on two datasets respectively. There are 4 (i.e., 2 × 2 = 4) tasks in total. You can download all necessary files from amlm (dataset files and model weights files are included in the `amlm/pickle` and `amlm/models` folders respectively), and you can utilize any information we provide to conduct effective membership inference attacks.

Datasets

We offer two datasets (i.e., CIFAR10 and Tiny ImageNet) to conduct the experiments. For each task, we offer the following three data files.

- Shadow dataset `shadow.p`
    - Used by the shadow model and the attack model. You can split it to train a shadow model and generate training data for the attack model
    - Each element is a Python List containing the image and the ground-truth label information
- Evaluation dataset `eval.p`
    - Used by the attack model and the target model. It contains some real members and non-members for the target model. You can evaluate the attack performance of your attack model on it before the final submission
    - Each element is a Python List containing the image, the ground-truth label, and the membership status (whether this image is a member in the training dataset of the target model, 1 for member and 0 for non-member) information
- Test dataset `test.p`
    - Used by the attack model and the target model. It also contains some real members and non-members for the target model. You should use your attack model to provide predictions (i.e., member or non-member) on `test.p` and submit your prediction results to us
    - Each element is a Python List containing the image and the ground-truth label information

Both `eval.p` and `test.p` contains some (possible) members for the target model. Note that, **we mainly judge the performance of your attack model based on the prediction results on the test dataset `test.p`**.

You can use the example code below to load the shadow dataset `shadow.p` or the test dataset `test.p` for the ResNet34 target model trained on the CIFAR10 dataset.

```
import pickle
import torch

DATA_PATH = 'amlm/pickle/cifar10/resnet34/shadow.p'
# Change the DATA_PATH to your local pickle file path

device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

with open(DATA_PATH, "rb") as f:
    dataset = pickle.load(f)

dataloader = torch.utils.data.DataLoader(
    dataset, batch_size=64, shuffle=False, num_workers=2)

for batch_idx, (img, label) in enumerate(dataloader):
    img = img.to(device)
    # other operations...
```

Note that, to train a shadow model and generate training data for the attack model, you can split the shadow dataset by yourself.

Similarly, you can use the following example code to load the evaluation dataset `eval.p`.

```
import pickle
import torch

DATA_PATH = 'amlm/pickle/cifar10/resnet34/eval.p'
# Change the DATA_PATH to your local pickle file path

device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

with open(DATA_PATH, "rb") as f:
    dataset = pickle.load(f)

dataloader = torch.utils.data.DataLoader(
    dataset, batch_size=64, shuffle=False, num_workers=2)

for batch_idx, (img, label, membership) in enumerate(dataloader):
    img = img.to(device)
    # other operations...
```

Models

You need to conduct the experiments on two target model architectures (i.e., ResNet34 and MobileNetV2). The pre-trained model weights are provided in the `aamlm/models` folder. You can use the example code below to load the model weights for the ResNet34 target model trained on the CIFAR10 dataset.

```
import torch
import torchvision.models as models

MODEL_PATH = 'amlm/models/resnet34_cifar10.pth'
# Change the MODEL_PATH to your local model path

device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

target_model = models.resnet34(num_classes=10).to(device)
# Change num_classes to 200 when you use the Tiny ImageNet dataset

state_dict = torch.load(MODEL_PATH, map_location=device)
target_model.load_state_dict(state_dict['net'])
# Test accuracy
acc = state_dict['acc']
# Training epoch (start from 0)
epoch = state_dict['epoch']
```

You can use any model architecture for the shadow model and the attack model.

Submission

You need to **submit your final prediction results on the test dataset `test.p` to hai.huang@cispa.de or yugeng.liu@cispa.de** to be checked by us. There should be **four** prediction result files in your submission. If you have successfully pass the test, you need to submit the code.

For each target model trained on each dataset, please save the prediction results on `test.p` into a binary file in NumPy. You can use the toy example below to store your results.

```
import numpy as np

# prediction is the attack model prediction with the type of NumPy array
containing all the test elements
# Each element of prediction is 1 (member) or 0 (non-member)

np.save('task0_resnet34_cifar10.npy', prediction)
```

Note that, please **do NOT change the order of the given testing samples**, otherwise it may negatively affect the evaluation results for your predictions on our side. Moreover, please name the prediction result files for different tasks as follows.

File Name	Task ID	Dataset	Model
`task0_resnet34_cifar10.npy`	0	CIFAR10	ResNet34
`task1_mobilenetv2_cifar10.npy`	1	CIFAR10	MobileNetV2
`task2_resnet34_tinyimagenet.npy`	2	Tiny ImageNet	ResNet34

File Name	Task ID	Dataset	Model
task3_mobilenetv2_tinyimagenet.npy	3	Tiny ImageNet	MobileNetV2

We strongly recommend that you evaluate the performance of your attack model on the evaluation dataset `eval.p` and get satisfying results (usually test accuracy larger than 60%) on it before submitting your final prediction results on the test dataset `test.p`. The expected test accuracy of the membership inference attack on `eval.p` for the aforementioned four tasks should be around 68%, 65%, 90%, and 80% respectively. Usually higher test accuracy is preferred, but you had better avoid the overfitting of your attack model. Finally, please also provide the names and student IDs of all group members in your email.