CIS 6261: Trustworthy Machine Learning (Spring 2023)
Homework 3 — Privacy Attacks on Machine Learning Models

Name: Anol Kurian Vadakkeparampil

April 10, 2023

This is an individual assignment. Academic integrity violations (i.e., cheating, plagiarism) will be reported to SCCR! The official CISE policy recommended for such offenses is a course grade of E. Additional sanctions may be imposed by SCCR such as marks on your permanent educational transcripts, dismissal or expulsion.

Reminder of the Honor Pledge: On all work submitted for credit by Students at the University of Florida, the following pledge is either required or implied: *"On my honor, I have neither given nor received unauthorized aid in doing this assignment."*

Instructions

Please read the instructions and questions carefully. Write your answers directly in the space provided. Compile the tex document and hand in the resulting PDF.

In this assignment you will implement and evaluate several membership inference attacks in Python. Use the code skeleton provided and submit the completed source file(s) alongside with the PDF.[1]

Assignment Files

The assignment archive contains the following Python source files:

- `hw.py`. This file is the main assignment source file.
- `nets.py`. This file defines the neural network architectures and some useful related functions.
- `attacks.py`. This file contains attack code used in the assignment.

Note: You are encouraged to take a look at the provided files. This may help you successfully complete the assignment.

[1] You should use Python3 with Tensorflow 2. You may use HiPerGator or your own system. If you use your own system you may need to install the required packages (e.g., numpy, scikit-learn, tensorflow, etc.). This assignment can be done with or without GPUs.

Problem 1: Training Neural Networks (20 pts)

For this problem, you will train neural networks to do MNIST classification. You will get familiar with the provided code skeleton and answer some questions about what you observe when running the code.

To run the code for this problem, use the following command.

```
python3 hw.py problem1 <nn_desc> <num_epoch>
```

Here `<nn_desc>` is a neural network description string (no whitespaces). It can take two forms: `simple,<num_hidden>,<l2_reg_const>` or `deep`. The latter specifies the deep neural network architecture (see `get_deeper_classifier()` in `nets.py` for details), whereas the former specifies a simple neural network architecture (see `get_simple_classifier()` in `nets.py` for details) with one hidden layer with `<num_hidden>` neurons and an L_2 regularization constant of `<l2_reg_const>`. Also, `<num_epoch>` is the number of epoch to train for.

For example, suppose you run the following command.

```
python3 hw.py problem1 simple,32,0.001 50
```

This will train the target model on 2000 MNIST data records (i.e., images) for 50 epochs. The target model architecture is a neural network with a single hidden layer of 32 neurons which uses L_2 regularization with a constant of 0.001.[2] (The loss function is the categorical cross entropy.)

1. (5 pts) Run the code using the command provided above. What is the training accuracy? What is the test accuracy? How overfitted is the target model you just trained?

   90.6% : Training Accuracy

   83.1% : Test accuracy

   The training accuracy is 90.6%, while the testing accuracy is 83.1%. This difference in accuracy indicates that the model is likely overfitting to the training data.

   Therefore, we can conclude that the target model is moderately overfitted.

2. (10 pts) Run the code to train the target model while varying the number of training epochs (from 10 to 1000), the number of hidden layer neurons (from 32 to 1024), and the regularization constant (from 0.0 to 0.01).

   What do you observe? For what combination of parameters is the train/test accuracy the highest? For each parameter you varied, explain its impact on accuracy and overfitting.

   python3 hw.py problem1 simple,1024,0.005 800

   Training Accuracy and Advantage : 99.4,0.11

   Testing Accuracy and Advantage : 90.0,0.41

   Highest accuracy rates were observed for 'simple,1024,0.005 800' these parameters.

   Training epochs, is the number of times the model gets to see the data during training. When we increase this number, the model has more chances to learn and improve its accuracy. However, as we keep increasing it too much, the model can start memorizing the training data instead of generalizing, which leads to overfitting and a decrease in accuracy on the test data.

   The number of neurons in the hidden layers, are responsible for doing the actual computation in the model. On increasing the number of neurons, the model has more capacity

[2] By default, for problem 1, the code will provide detailed output about the training process and the accuracy of the target model.

to learn complex patterns in the data and can improve its accuracy. But, just like with training epochs, on adding too many neurons, the model can become too complex and start overfitting the data, which leads to a decrease in accuracy on the test data.

The regularization constant, is a penalty we impose on the model for having large weights. It helps prevent overfitting by making the model simpler and less prone to memorizing the training data. However, on setting the regularization constant too high, the model becomes too simple and is not be able to learn the relationships in the data, which leads to underfitting and lower accuracy.

3. (5 pts) Take the best performing architecture you found above. For this question, we are interested in inputs that are misclassified by the target network. Use the provided `plot_images()` function to plot some examples of test images that are misclassified. Paste the plot below. Do you notice anything?

It seems like the model misclassifies certain images that betray the regular pattern which is to be expected. We can conclude that the model is generalising well based on wrong classification of these outliers.

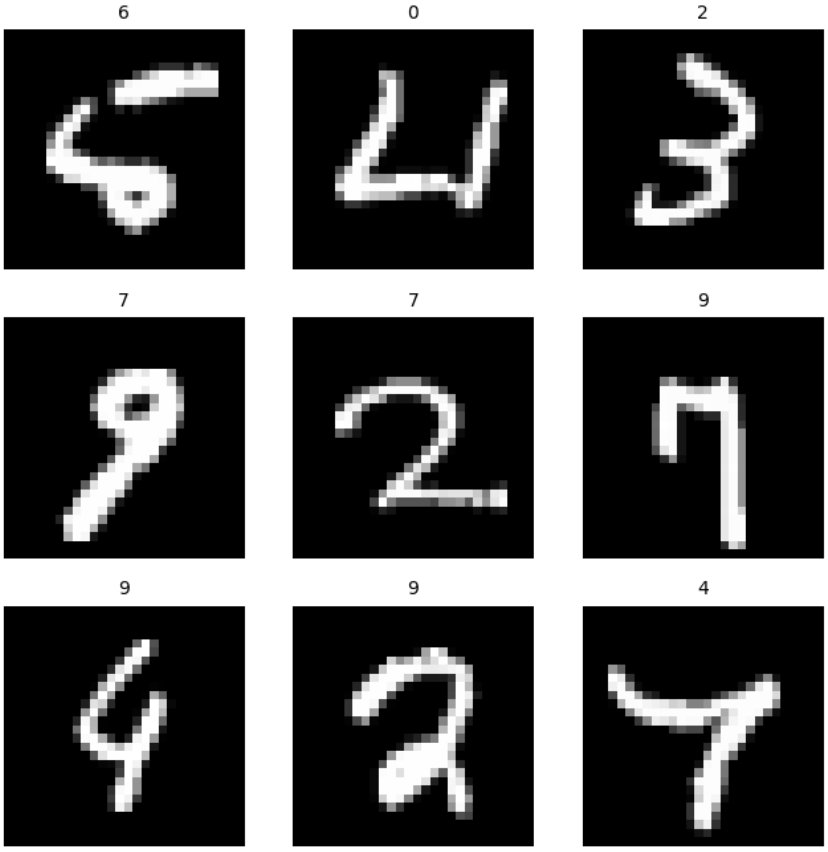

Figure 1: Plot with predictions and images

Problem 2: Implementing the Shokri et al. Attack (30 pts)

For this problem you will implement the Shokri et al. [1] attack. Put your code in the `shokri_attack_models()` function which is located in `attacks.py`. The companion function `do_shokri_attack()` is already provided so you do not have to implement it.

Refer to the course slides and/or the paper [1] for instructions on how to implement the attack. Comments in the provided code skeleton are here to guide your implementation.

To run the code for this problem, use the following.

```
python3 hw.py problem2 <nn_desc> <num_epoch> <num_shadow> <attack_model>
```

Here `<num_shadow>` is the number of shadow models to use and `<attack_model>` is a string denoting the attack model type (scikit-learn). See the code for options.

Answer the following questions.

1. (10 pts) Once you have implemented the attack, run the following command:

   ```
   python3 hw.py problem2 simple,1024,0.0 50 10 LR
   ```

   Note: you may need to re-train the model (see problem 1).

   What is the accuracy of the attack? What is the advantage?

   Accuracy : 70.8% — Advantage : 0.42

2. (10 pts) Run the attack on the most overfitted target model (according to the parameters you found in the previous problems and your overfitting metric). Also run the attack on the least overfitted but best performing target model. Compare the attack accuracy and advantage in both cases. What do you notice? According to your experiments what are the factors that lead to attack success? (Justify your answer.)

   Overfitted Model : simple,32,0.01 800

   Training Accuracy and Advantage: 97.8,0.65 — Testing Accuracy and Advantage : 85.2,0.95

   Shokri attack (LR) accuracy, advantage: 65.3%, 0.31

   ———————————————————————————————

   Best performing model : Command: python3 hw.py problem1 simple,1024,0.005 800

   Training Accuracy and Advantage: 99.4,0.11 — Testing Accuracy and Advantage : 90.0,0.41

   Shokri attack (LR) accuracy, advantage: 68.8%, 0.38

   ———————————————————————————————

   When we compare the two models, we see that the best performing model is more resistant to the Shokri attack than the overfitted model. The attack accuracy and advantage are higher in the best performing model than in the overfitted model. This is a positive sign as it shows that the best performing model is more robust to adversarial examples than the overfitted model.

   The factors that determine the success of an attack are related to how well the model can learn from data and how vulnerable it is to adversarial examples. In the case of the overfitted model, it has memorized the training data and is not able to generalize well to new examples. This makes it more vulnerable to adversarial examples, which

are carefully crafted examples that can fool the model into making incorrect predictions. On the other hand, the best performing model has been regularized and has learned to generalize better, which makes it less vulnerable to adversarial examples and more robust to attacks.

3. (10 pts) Now vary the attack model type. What do you notice? What are the best/worst performing attack model types?

(Overfitted) simple,32,0.01 800 10

Shokri attack (SVM) accuracy, advantage: 65.6%, 0.31

Shokri attack (DT) accuracy, advantage: 54.2%, 0.08

Shokri attack (RF) accuracy, advantage: 59.1%, 0.18

Shokri attack (NB) accuracy, advantage: 62.6%, 0.25

Shokri attack (MLP) accuracy, advantage: 64.3%, 0.29

———————————————————————————

(Best Performance) simple,1024,0.005 800 10

Shokri attack (SVM) accuracy, advantage: 68.2%, 0.36

Shokri attack (DT) accuracy, advantage: 61.9%, 0.24

Shokri attack (RF) accuracy, advantage: 68.5%, 0.37

Shokri attack (NB) accuracy, advantage: 63.7%, 0.27

Shokri attack (MLP) accuracy, advantage: 71.2%, 0.42

———————————————————————————

Upon analyzing the results, we notice that the best performing model has higher accuracy and advantage than the overfitted model for all types of attack models.

Looking at the results, it seems like the MLP attack model is the most effective in both cases, with the highest accuracy and advantage compared to the other models. On the other hand, the decision tree attack model seems to be the least effective, with the lowest accuracy and advantage in both cases. The SVM and random forest attack models have a moderate performance, with accuracy and advantage values somewhere in between the best and worst performing models. For the Naive Bayes attack model, it shows moderate performance in the best performing model, but its performance is relatively weaker in the overfitted model.

Problem 3: More membership inference attacks (30 pts)

For this problem you will implement three more attacks.

- Loss attack (`do_loss_attack()`): for this attack you are given the mean and standard deviation of the target model's training loss and testing loss. You can assume that the loss follows a Gaussian distribution.

  The attack works as follows: given a target record, measure its loss on the target model and then decide (IN or OUT) based on which loss distribution the sample most likely comes from.

- Loss attack2 (`do_loss_attack2()`): for this attack you are given only the mean and standard deviation of the target model's training loss. In addition, the attack function takes a threshold parameter. Again, you can assume that the loss follows a Gaussian distribution.

  The attack works as follows: given a target record, measure its loss on the target model and then decide (IN or OUT) based on whether the probability of obtaining a loss value as extreme or less extreme than that is *lower* than the threshold.

- Posterior attack (`do_posterior_attack()`): for this attack you only have the ability to query the target model. You are also given a threshold parameter.

  The attack works as follows: given a target record, query it on the target model to obtain its posterior (i.e., the predicted probability over the true class label) and then decide (IN or OUT) based on whether the posterior is *greater* than the threshold.

Implement all three attacks and answer the following questions.

1. (5 pts) Pick a set of parameters. What is the attack accuracy and advantage of the loss attack? Specify the command you ran!

   'python3 hw.py problem3 simple,1024,0.005'

   Loss attack accuracy, advantage: 60.4%, 0.21

2. (10 pts) Loss attack2 takes a threshold parameter. What is the optimal threshold value? Justify your answer. (Hint: you can write some code to find out experimentally.)

   The best threshold for loss attack 2 is: 0.7

   loss 2 acc: [0.5, 0.5, 0.7, 0.65, 0.62] for threshold values [0.1,0.3,0.5,0.7,0.9].

   We iterated over the threshold values and found the optimal one..

3. (10 pts) The posterior attack takes a threshold parameter. What is the optimal threshold value? Justify your answer. (Hint: you can write some code to find out experimentally.)

   Posterior attack accuracy, threshold: 47.0%, 0.10

   Posterior acc: [0.47, 0.44, 0.42, 0.4, 0.34] for threshold values [0.1,0.3,0.5,0.7,0.9].

   We iterated over the threshold values and found the optimal one..

4. (5 pts) Which of the three attacks performs best?

The Loss attack 2 performs best among the three attacks, with an accuracy of 60.4% and an advantage of 0.21. The optimal threshold value for the Loss attack 2 is 0.5, which corresponds to a high accuracy value of 0.7.

On the other hand, the Posterior attack has the lowest accuracy among the three attacks, with an accuracy of 47.0% and a threshold value of 0.1. The accuracy values obtained for different threshold values in the Posterior attack are [0.47, 0.44, 0.42, 0.4, 0.34].

Therefore, the Loss attack 2 performs better than the other two attacks in terms of accuracy and advantage.

Problem 4: Overfitting & Other factors (20 pts)

For this problem, we are interested in understanding how much of a role overfitting plays in the success of the attack (compared to other factors).

1. (10 pts) Play with the neural network architecture and other parameters to train target models with varying level of overfitting. In each case, run all four attacks and record the accuracy and advantage. Paste the results below as plot(s) or table(s). What do you observe?

Table 1: Model Comparison

Model	**Shokri (Acc, Adv)**	**Loss (Acc, Adv)**	**Loss2 (Acc, Adv)**	**Posterior (Acc, Adv)**
simple,32,0.01 100	64.1%, 0.28	62.0%, 0.24	65.6%, 0.31	34.9%, -0.30
simple,32,0.01 800	64.6%, 0.29	62.0%, 0.24	65.6%, 0.31	34.9%, -0.30
simple,256,0.01 600	69.0%, 0.38	62.4%, 0.25	68.5%, 0.37	36.4%, -0.27
simple,1024,0.005 800	70.5%, 0.41	60.4%, 0.21	70.2%, 0.40	39.5%, -0.21
simple,1024,0.005 1000	70.5%, 0.41	60.4%, 0.21	70.2%, 0.40	39.5%, -0.21
deep 600	66.2%, 0.32	59.4%, 0.19	65.0%, 0.3	38.6%, -0.23
deep 800	67.0%, 0.34	58.4%, 0.17	64.0%, 0.28	39.5%, -0.21

Can you find significantly different architectures / sets of parameters with a similar overfitting level but with different attack success? What do you conclude?

(num_shadows = 10)

Manually found significantly different architectures / sets of parameters with a similar overfitting level from output through training problem 1 iteratively over varying parameters -

———————————————————————————

simple,64,0.01 900 — Train accuracy : 98.7, Test accuracy : 87.6

Shokri attack (MLP) accuracy, advantage: 67.1%, 0.34

Loss attack accuracy, advantage: 63.3%, 0.27

Loss attack2 accuracy, advantage: 67.4%, 0.35

Posterior accuracy, advantage: 34.1%, -0.32

———————————————————————————

simple,256,0.001 300 — Train accuracy : 98.7, Test accuracy : 87.6

Shokri attack (MLP) accuracy, advantage: 69.0%, 0.38

Loss attack accuracy, advantage: 61.4%, 0.23

Loss attack2 accuracy, advantage: 67.5%, 0.35

Posterior accuracy, advantage: 37.0%, -0.26

———————————————————————————

Overfitting occurs when the model learns the training data too well and becomes less able to generalize to new, unseen data. Attack success, on the other hand, refers to the model's ability to correctly classify adversarial examples.

It is possible that a model with a more complex architecture or a larger set of parameters may have a similar level of overfitting to a simpler model, but it may perform better or worse on adversarial examples. This is because the model's ability to generalize to unseen data and its robustness to attacks are two different aspects of its performance.

2. (10 pts) Now modify `nets.py` to add regularization using dropout. (Make sure the model accuracy does not decrease too much.) Once you have done this, re-run the attacks with different dropout rates. What do you observe?

(num_shadows = 10)

simple,64,0.01 900

———————————————————————————————

Rate 0.2:

Shokri attack (MLP) accuracy, advantage: 67.8%, 0.36

Loss attack accuracy, advantage: 63.3%, 0.27

Loss attack2 accuracy, advantage: 67.4%, 0.35

Posterior accuracy, advantage: 34.1%, -0.32

———————————————————————————————

Rate 0.4:

Shokri attack (MLP) accuracy, advantage: 65.8%, 0.32

Loss attack accuracy, advantage: 62.7%, 0.26

Loss attack2 accuracy, advantage: 66.8%, 0.32

Posterior accuracy, advantage: 35.0%, -0.20

=======================================

simple,256,0.001 300

———————————————————————————————

Rate 0.2:

Shokri attack (MLP) accuracy, advantage: 69.4%, 0.39

Loss attack accuracy, advantage: 61.4%, 0.23

Loss attack2 accuracy, advantage: 67.5%, 0.35

Posterior accuracy, advantage: 37.0%, -0.26

———————————————————————————————

Rate 0.4:

Shokri attack (MLP) accuracy, advantage: 68.5%, 0.35

Loss attack accuracy, advantage: 60.1%, 0.19

Loss attack2 accuracy, advantage: 66.5%, 0.30

Posterior accuracy, advantage: 39.3%, -0.17

=======================================

On increasing the model's dropout rate from 0.2 to 0.4, we see that the model becomes more robust to adversarial attacks, resulting in lower accuracy and advantage for the Shokri attack. The model's regularization would also increase, which led to decreased

accuracy and advantage for the loss attack and loss attack2. However, the posterior attack accuracy and advantage increased as the model produces more diverse posterior distributions when it is less certain about its predictions.

References

[1] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), IEEE, pp. 3–18.