EL-GY-9133 Machine Learning for Cyber-Security

**Lab 3: Inference Attacks on Deep Neural Networks**

*Release Date*: 04/10/2018; *Due Date*: Midnight, 04/20/2018

**Overview**

In this lab, you will investigate training data inference attacks on Deep Neural Networks using the MNIST digits dataset as a benchmark. You will then evaluate differentially private machine learning as a defense against adversarial perturbations.

**Dataset**

The MNIST dataset is a commonly used “toy” benchmarks for machine learning. It contains 28X28 grayscale images of hand-drawn digits from 0-9, along with the associated labels. The dataset is available as part of the *tensorflow* package, which you will be using extensively in this lab. Please see the sample Python code for this homework for details on how to read in this dataset.

**What You Have to Do**

Starting with the **baseline DNN** that we provided for Assignment 2 (recall that the DNN has a 784 (28x28) dimensional input, a 10-dimensional output and one hidden layer with 300 hidden neurons and ReLU activations) you will first implement attacks that use the “Activation Maximization” (AM) method to reverse engineer representative training data samples for each class.

For each class [0,9], AM starts with an initial guess of what a representative image for that class looks like. Starting with this initial guess, AM uses gradient ascent to increase the prediction probability for class (or equivalently uses gradient descent to minimize the loss of the network for target label ). The algorithm terminates when it reaches an input for which no further increase in in the prediction probability for class is observed. This process is repeated for each .

1. **AM with Random Initialization:** Implement AM with random initialization. That is, for each start the AM algorithm with a random image. The amplitude of each pixel of the random image is picked uniformly at random between [0,255] and then scaled to lie between [0,1]. Repeat for all [0,9] each time with a different random guess. For each [0,9], determine the RMSE error between the recovered image for that class and the image in the training dataset closest to the recovered image. The final error is defined as the average RMSE error over all 10 classes.
2. **AM with Mean Initialization:** Same as above except that AM is initialized with an image that is the average over *all* training data images. Repeat for all [0,9]; in each run you will start with the same initial guess. Compute the RMSE error for each class and the final error as above.
3. **AM with Per-Class Mean Initialization:** Same as above, except when running AM for class , start with an initial guess equal to the mean of all images in the training set from class . Repeat for all [0,9]. Compute the RMSE error for each class and the final error as above.
4. **Differentially Private Training:** Finally, you will study the efficacy of DP mechanisms to defend against training data inference. We will use the black-box DP mechanism, i.e., we will add zero mean Laplacian noise to the trained neural network weights. The probability density function of the zero mean Laplace distribution is controlled by a single parameter . For each trained weight, you will sample independently from the Laplacian distribution and add the obtained random value to the weight. You can read more about picking from a Laplacian distribution here: <https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.laplace.html>

To understand the trade-off between accuracy and privacy, try the following values of where and is the standard deviation of the weights of the neural network.

For each value of determine:

1. The accuracy of the new model with noise added on the test set
2. Re-run the AM with Per-Class Mean Initialization algorithm on the new model. For each [0,9], determine the RMSE error between the recovered image for that class and the image in the training dataset closest to the recovered image. The final error is defined as the average RMSE error over all 10 classes.

**What to Submit**

1. A report that describes the methodology you used to implement the AM algorithm. Note that the problem is open-ended in that you can play with the step size of the AM algorithm, and potentially even use an adaptive step-size.
2. For Parts 1-3, plot the recovered images for each class (10 per problem) and the computed RMSE for that image. Also indicate the final error.
3. For part 4, tabulate the prediction accuracy versus the final error for each value of Also indicate the value of used.

1. Your Python code along with any instructions required to execute the code. Details on how to submit your code will be provided on NYU Classes.