Trustworthy Machine Learning, Spring 2022 *Mahmood Sharif*

Homework Assignment 1

Version No.: 1

Introduction

In this assignment, you will practice background material by implementing basic classifiers (namely, logistic regression and fully connected neural networks) and training them to classify data from the MNIST dataset. Subsequently, you will implement white-box attacks to produce adversarial examples that mislead the classifiers.

If you have limited prior experience with logistic regression and neural networks, or need to refresh your memory, we recommend that you read (links are embedded into the text):

- Stanford's CS231's course material on linear classification, optimization, and backpropagation.
- The introductory chapters of Goodfellow et al.'s Deep Learning book.

Environment

To work on the assignment, you need to install two Python packages: `numpy` and `Pillow`. To ensure compatibility with our environment and avoid unexpected issues, we recommend using Python 3.8, and installing `numpy` version 1.22.2 and `Pillow` version 9.0.1. You may want to consider creating a virtual Python environment specifically for this assignment using `virtualenv`.

Submission

1. The deadline for submitting solutions is **March 20th, at 23:59**.
2. You handin should include the following:
   - A writeup named `writeup.pdf`.
   - The source code, including all `.py` files with the blanks filled. With the exception of the original calls to print that are included in the code (please avoid changing the formatting), make sure that your code *does not* print to stdout.
   - Place your writeup and code under a directory titled `<ID>/`, create a compressed tar file by running `tar -czvf <ID>.tgz <ID>/`, and upload `<ID>.tgz` to Moodle.

Question #1 - Exploratory data analysis (EDA) (5 points)

As a recommended practice, one should first examine the data before training ML models. Familiarize yourself with the data stored in `data.npz` (you can find code for reading the data in `utils.py`), and answer the following questions:

- How many samples, overall, are included in the dataset?

- How many classes do the data belong to? What are these classes?
- What is the dimensionality of the data?
- What are the sizes of the training, validation, and test sets? How many samples of each class are included in each set?

Save the code you wrote for the EDA in `main_a.py`.

Question #2 - Logistic regression (30 points)

Here you will implement two logistic regression models, one for binary classification-tasks, and another for multi-class classification. The binary classifier you will train will receive digit images as input and output the parity of the digit (see `main_b.py`). In contrast, the multi-class model aims to predict the exact digit (see `main_c.py`). Complete the missing code in `logistic_regression.py` to implement the classifiers, and answer the following:

1. What is the test accuracy of each classifier?
2. Store the weights of the matrix W as images and examine them (check `utils.py` for a potentially useful function). Can you interpret how the classifiers work based on the visualizations? What are the foreground features that matter? Does the background play a role in the classification?

Question #3 - Fully connected neural networks (30 points)

In this question, you will train two fully connected artificial neural networks to solve the same classification problems tackled in the previous question (see `main_d` and `main_e.py`). To train the networks, you need to fill out the missing code in `neural_networks.py`, implementing the necessary code for computing forward passes, losses, and backward passes. What is the test accuracy achieved by each of the networks?

Question #4 - White-box attacks (35 points)

Next, you will implement two commonly used attacks that we saw in class—FGSM and PGD. Fill-out the missing pieces in `attacks.py` to implement untargeted and targeted variants of both attack methods. Subsequently, run the code in `main_f.py` and `main_g.py` to evaluate the success rates of attacks against the model trained in `main_e.py`. Which of FGSM and PGD was more successful?

As a means to avoid errors in attack implementation (e.g., perturbing the input more that the attack budget), it is recommended to visualize adversarial examples, and add assertions to check that the L_p-norm constraints are satisfied. Add such assertions to the code, visualize five adversarial examples for each attack type along with their benign counterparts, and report the target in the case of targeted attacks.

You can find the weights of a pre-trained model stored under `models/pretrained.npz`. This pre-trained model has the same architecture as the model trained in `main_e.py`. Load the model, and evaluate the success rate of untargeted FGSM against it. Was FGSM less successful than before? Why did this happen? Can you think of a way to improve the success rate of FGSM? Implement the improvement and report how it affects the attack success rate.