Adversarial Attacks

Question 1: TRADES by Hand

[7 marks] Consider the model $f(x) = \sigma(wx)$, where x is the input, σ is the logistic function, and w is a scalar weight. The output of this network represents the probability that the class of the input is 1 (as opposed to 0).

You can evaluate the performance of your model on any given sample, (x, t), where $x \in \mathbb{R}$, and $t \in [0, 1]$ using the binary cross-entropy (BCE) loss function,

$$L(y, t) = -t \ln y - (1 - t) \ln(1 - y)$$

where $y = f(x)$.

You are given the input $x = 1$, with corresponding target class $t = 1$. Suppose the current value of w is 2, and you want to use this new data point to update that connection weight.

A. Compute a single step of gradient descent on w, using a step multiplier of 0.1.

$$w \leftarrow w - 0.1 \frac{\partial L}{\partial w}$$

Show your work, and clearly indicate what your updated value is for w.

B. Now you will calculate a different gradient descent step for w, this time using the TRADES algorithm. Again, assume w is initialized to 2, and follow these steps:

(a) Derive the adversarial gradient, $\frac{\partial L}{\partial x}$, for the sample $(x, t) = (1, 1)$. Show your work, and clearly indicate the value of $\frac{\partial L}{\partial x}$.

(b) Using the derivative from (a), create an adversarial input, $\tilde{x}$, by taking a single gradient-**ascent** step (to increase the BCE loss) using a step multiplier of 0.5. Show your work, and clearly indicate your value for $\tilde{x}$.

(c) Compute a single step of gradient descent on w, using a step multiplier of 0.1, but using the TRADES loss

$$\mathcal{L} = L(y, t) + L(y, \tilde{y})$$

where $\tilde{y} = f(\tilde{x})$ is the output for the adversarial input $\tilde{x}$. Clearly show your derivation of $\frac{\partial \mathcal{L}}{\partial w}$, and indicate the updated value of w.

Question 2: FGSM

Download the jupyter notebook `a09_YOU.ipynb`. This notebook also requires PyTorch (`torch`), `tqdm`, and the MNIST dataset. You may also want to download the file that contains a set of pre-trained network parameters, `mnist_trained.pt`; the notebook includes code to load it and use it.

For this question, you will implement the Fast Gradient Sign Method (FGSM), a simple, gradient-based adversarial attack on a network trained on MNIST. The notebook includes a class called `MyNet`. It can also load an instance of that network class from the file `mnist_trained.pt` which has already been trained on MNIST.

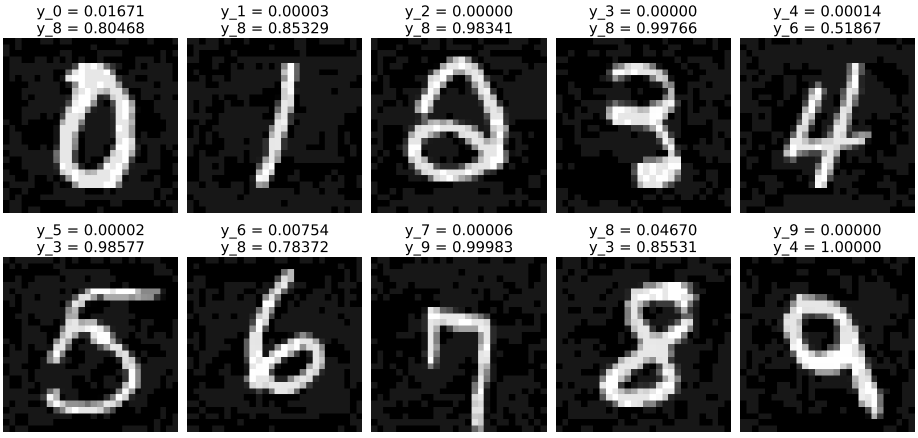
A. **fgsm:** [5 marks] Complete the function `fgsm`, which implements the Fast Gradient Sign Method.

> **Fast Gradient Sign Method (FGSM)**
> Given input x and class t, create an adversarial input $\tilde{x}$ using
>
> $$\tilde{x} = x \pm \varepsilon \operatorname{sign}\Big(\nabla_x E\big(f(x), t\big)\Big)$$
>
> The sign in front of ε depends on whether the attack is targetted or untargetted.

See the docstring for details on how it is supposed to work. Your function must work for both targetted and untargetted adversarial attacks. Note that the FGSM method generates an adversarial input such that $\|x - \tilde{x}\|_\infty \leq \varepsilon$.

B. **Untargetted Adversarial Attack:** [7 marks] Choose one of each digit (from 0 to 9) from the MNIST dataset and ensure that each digit sample is **correctly classified** by the network.

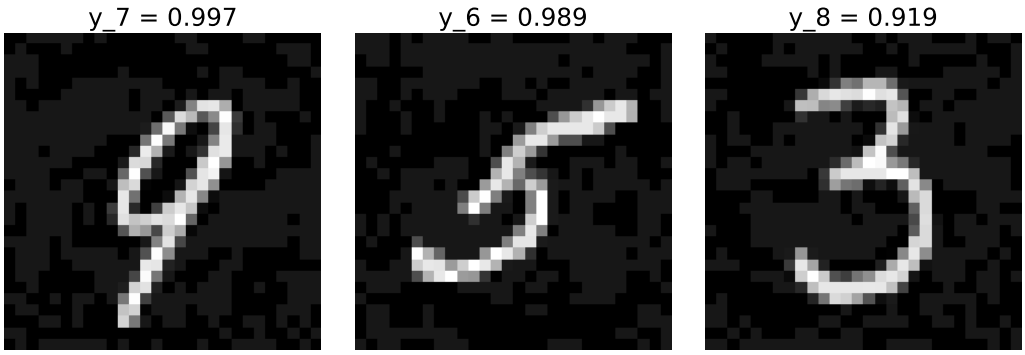
For each digit, run your `fgsm` function on it using $\varepsilon = 0.05$, and display the results of how the network interprets the adversarial image. For each adversarial digit, print the **original class** of the digit, the **network's output (probability) for that original class**, the network's **predicted class for the adversarial image**, and the **output (probability) for that class**. Also show the **adversarial images** in order from 0 to 9 (see the figure below).



Note: Your method does not need to misclassify every digit all the time. Some input samples are less susceptible to attack than others.

C. **Targeted Adversarial Attack:** [3 marks] Run your `fgsm` method on a selection of 3 different inputs (from different classes) using an ε value of 0.05. But this time, run a targetted adversarial attack. For each of the 3 inputs, choose a target class different from its true class. Display each **adversarial**

image, along with the **target class**, and the network's output **probability for that target class**, as shown in the figure below.



What to submit

Question 1: You can:

- typeset your solutions using a word-processing application, such as Microsoft Word, LaTeX, Google docs, etc., or
- write your solutions using a tablet computer, or
- write your solutions on paper and take photographs.

In any case, it is **your responsibility** to make sure that your solutions are sufficiently legible.

Submit your solution to Crowdmark only. Crowdwark will accept PDFs or image files (JPEG or PNG). You may submit multiple files for each question, if needed.

Question 2: You must submit your jupyter notebook in two places: **Crowdmark**, and **D2L**.

Crowdmark: Export your jupyter notebook as a PDF, and submit the PDF to Crowdmark.

D2L: Submit your `ipynb` file to Desire2Learn in the designated dropbox. You do not need to zip the file.

Make sure you submit your solutions, and not just a copy of the supplied skeleton code. We suggest you rename the `ipynb` file by replacing "`YOU`" with your WatIAM ID (eg. `a09_jorchard.ipynb`).

Important note about submitting code
We re-run your notebook in its entirety. So make sure that all your code cells run, from top to bottom. Any code cell that crashes will stop us from re-running your notebook (and you don't want that). Also, we extract the class and function definitions and autotest them. To facilitate this extraction process, please ensure that:

- all `import` commands are in a single code cell (at the top of the notebook), and
- code cells that contain function definitions and/or class definitions should not include any lines of code that are not part of the definition.