Create a Jupyter notebook for this assignment, and use Python 3. Write documented, readable and clear code (e.g. use reasonable variable names). Submit this notebook interspersing any textual answers in markdown cells (using LaTeX), clearly labeled, along with your code.

1. An important part of training against adversarial is finding adversarial examples. To do this for every x in our data set such that $x \in \mathbb{R}^d$ we create an adversarial counterpart x_{adv} to create a set of x_{adv} that we can test against the trained neural network. The goal of x_{adv} is to have the same label as x but be purposefully designed to be misclassified by the neural network. To find the x_{adv} for any given x we use the following equation:

$$x_{adv} = x + argmax_{\delta \in S} L(w, x + \delta, y)$$

Where $S \subseteq \mathbb{R}^d$, L is the loss function, w is the weights determined by training the non-adversarial set of x and y as the corresponding correct label for x and therefore the corresponding correct label for x_{adv}. For the purposes of our model, the S for any given x will be the l_∞ ball around that x. Explain the geometric intuition behind why finding the argument for the maximum loss function in S would give us a good adversarial example.

For our model, we will use the iterative algorithm:

$$x^{t+1} = P_{x+S}(x^t + a * sign(\nabla_x L(w, x, y)))$$

We can notice that this is just projected gradient descent for the negative loss function projecting on the set $x + S$. We already know that projected gradient descent minimizes the input for a function within a certain convex set. Minimizing the negative loss function is the same as maximizing the log function. Thus, we are finding the input within the set $x + S$ that maximizes the loss function which is exactly $x + argmax_{\delta \in S} L(w, x + \delta, y)$

2. Using the knowledge from the first problem, if we wanted to create a neural network that would defend against adversarial attacks, what modification should we add to defend against it during training(hint: what minimization problem do you need to change?)

3. Using the knowledge from problem 1, explain why increasing the capacity of the neural network and training with non-adversarial examples helps better classify adversarial sets.

4. Try the Adversarial Neural Network and the PGD attack yourself:

- Download the Adversarial Neural Network and PGD Attack at https://github.com/thomashopkins32/PGDAdversarialLearning.git and create your Jupyter Notebook in the base directory of this repo
- Download the Fashion-MNIST data set at https://github.com/zalandoresearch/fashion-mnist and put it in the PGDAdversarialLearning base repo

- Load the training and testing splits of the data set according to the instructions for loading the Python given in the README; use the same variable names. (Note, import mnist_reader from mnist_fashion/utils rather than from utils because mnist_fashion is no longer the base directory)
- Preprocess the training data so that all 784 features (pixel values) look essentially like standard Gaussians (this helps with accuracy and convergence). Do this by fitting an sklearn StandardScaler on the training data and applying it to the training and test data sets; see https://scikit-learn.org/stable/modules/preprocessing.html. Overwrite the original training and test data sets with these processed data sets
- Create an instance of LinfPGDAttack from pgf_attack.py using LeNet5(100).double() as the model. You can import LeNet5 from model.py (for the parameters epsilon, num_steps, step_size and random_start, you can find the values in the config file and simply use config['name'] to retrieve them). Then run perturb in order to obtain your adversarial x. Do this for both the training and test data
- Run LeNet5 on the non-adversarial training x. Then using the weights obtained from running LeNet5 on non-adversarial data, predict the adversarial training x and the adversarial test x. Report the confusion matrix.
- Run the modified neural network on the non-adversarial training x. Then using the weights obtained from running the modified neural network on non-adversarial data, predict the adversarial training x and the adversarial test x. Report the confusion matrix.
- What conclusions do you draw about the performance of the adversarial model compared to the non-adversarial model using the confusion matrices of the test set?