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. Show that it is possible to solve the inner maximization problem by reducing it:

$$max_{||\delta||_\infty \leq \epsilon} l(w^T(x+\delta), y)$$

where the loss function is $l(x, y) = log(1+exp(-y*x)) = L(y*x)$ (use the denotation of L for convenience, this will help!) which mean the actual minimization problem is:

$$max_{||\delta||_\infty \leq \epsilon} L(y*(w^T(x+\delta)+b)$$

For this problem $y \in \{+1, -1\}$.
You will notice that this means that we can reduce the outer minimization and inner maximization problem to a simple minimization problem.

2. You will evaluate empirically the impact of using an adversarial trained neural network:

- Download our files of PGDAdverserialLearning from this link: https://github.com/thomashopkins32/PGDAdversarialLearning. Create your Jupyter notebook in the "scripts" folder.
- Import LeNet5 from the file model in the scripts folder. This will accept a 28x28 sized input, flattens them into vectors and passes through a hidden layer with 84 neurons and the tanh activation layer. The output layer returns a number of logits equal to the class number.
- Determine if a gpu is available, and set a variable device accordingly. Load the FashionMNIST train and validation (test) datasets using the ToTensor transformation and create dataloaders for each, using per-epoch shuffling and a batch size of 128.
- Import the train file from scripts folder. This will contain the functions train_one_epoch(model, loss_func, optimizer, dataloader, attack, device='cpu') that takes the model, loss function, optimizer, data as well as an adversarial attack and evaluate(model, loss_func, dataloader, attack, device='cpu') that evaluates the model on a full testset.
- Create 2 networks with 10 in the parameter (this is the number of classes), one that is trained using adversarial examples and one that is trained through normal examples.
- Import LinfPGDAttack from the pgd_attack file and then create 2 instance of LinfPGDAttack (one for the adversarial model and one for the normal model) with the paramaters: model=the LeNet5 model you created (adversarial for the adversarial attack, normal for the normal attack),

epsilon=0.0313725, num_steps = 5, step_size = 2.0, random_start = true and device=whatever device you have set.

- Create Adam optimizers for each of the networks, using learning rate .01. Train each model for 20 epochs with cross-entropy loss.. Use the train_one_epoch function with attack=the adversarial attack for training the adversarial model and the train_one_epoch function with attack=None for the normal model. For each epoch, record the validation accuracy and loss. Then run the attack (adversarial attack for the adverserial training, normal attack for normal training) on the validation set and then evaluate the accuracy and loss for the validation set with adverserial attack and record those.

- Plot the validation accuracy and loss for the adversarial model (on separate graphs). For each graph, also plot the validation accuracy and loss for the validation set that we had run the adversarial attack through.

- Plot the validation accuracy and loss for the normal model (on separate graphs). For each graph, also plot the validation accuracy and loss for the validation set that we had run the adversarial attack through.

- Comment on your observations.