ECE 590-03: Homework #5
Adversarial Attacks and Defenses

Hai Li

ECE Department, Duke University — October 15, 2020

Objectives

Homework #5 covers the contents of Lectures 15 $\sim$ 17. This assignment starts by implementing several basic gradient-based adversarial attacks and analyzing how the ϵ of the attack influences the perceptibility of the noise. Then, you will evaluate the attacks in both the whitebox and blackbox settings, measuring the trade-off between attack success and ϵ in each. Finally, you will adversarially train some robust models and measure their ability to defend against such attacks.

Warning: You are asked to complete the assignment independently.

This lab has a total of **100** points plus 10 bonus points, yet your final score cannot exceed 100 points. You must submit your report in PDF format and your original codes for the lab questions through **Sakai** before **11:55:00pm, Thursday, October 29**. You need to submit **two individual files** including (1) *a self-contained report in PDF format* that provides answers to all the Lab questions and clearly demonstrates all your lab results and observations, (2) the *attacks.py* file which contains your attack implementations. **Note that 10 percent of the grade will be deducted for the submissions uploaded in a zip file.**

1 Lab 1: Environment Setup and Attack Implementation (20 pts)

In this section, you will train two basic classifier models on the `FashionMNIST` dataset and implement a few popular adversarial attack methods. The goal is to prepare an "environment" for attacking in the following sections and to understand how the adversarial attack's ϵ value influences the perceptibility of the noise. All code for this set of questions will be in the "Model Training" section of `HWK5_main.ipynb` and in the accompanying `attacks.py` file. Please include all of your results, figures and observations into your PDF report.

Lab 1 (20 points)

(a) (4 pts) Train the given *NetA* and *NetB* models on the `FashionMNIST` dataset. Use the default training parameters and save two checkpoints: "netA_standard.pt" and "netB_standard.pt". What is the final test accuracy of each model? Do both models have the same architecture? (Hint: accuracy should be around 92% for both models).

(b) (8 pts) Implement the ℓ_∞-constrained Projected Gradient Descent (PGD) adversarial attack in the `attacks.py` file. In the report, describe what each of the input arguments is controlling. Then, using the "Visualize some perturbed samples" cell in `HWK5_main.ipynb`, run your PGD attack using *NetA* as the base classifier and plot some perturbed samples using ϵ values in the range [0.0, 0.3]. At about what ϵ does the noise start to become perceptible/noticeable? Do you think that you (or any human) would still be able to correctly predict samples at this ϵ value? Finally, to test one important edge case, show that at $\epsilon = 0$ the computed adversarial example is identical to the original input image.

(c) (8 pts) Implement the Fast Gradient Sign Method (FGSM) attack in the `attacks.py` file. (Hint: treat the FGSM function as a wrapper around the PGD function). Then, plot some perturbed samples using the same ϵ levels from the previous question and comment on the perceptibility of the FGSM noise. Does the FGSM and PGD noise appear visually similar?

2 Lab 2: Measuring Attack Success Rate (40 pts)

In this section, you will measure the effectiveness of your PGD and FGSM attacks. Remember, the goal of an adversarial attacker is to perturb the input data such that the classifier outputs a wrong prediction, while the noise is minimally perceptible to a human observer. All code for this set of questions will be in the "Test Attacks" section of `HWK5_main.ipynb` and in the accompanying `attacks.py` file. Please include all of your results, figures and observations into your PDF report.

Lab 2 (40 points)

(a) (5 pts) Briefly describe the difference between a whitebox and blackbox adversarial attack. Also, what is it called when we generate attacks on one model and input them into another model that has been trained on the same dataset?

(b) (10 pts) *Whitebox Attack* - Using your pre-trained "NetA" as the whitebox model, measure the whitebox classifier's accuracy versus attack epsilon for both the FGSM and PGD attacks. Test at least eleven ϵ values across the range [0, 0.1] (e.g., `np.linspace(0,0.1,11)`) and plot the accuracy vs epsilon curve. For the PGD attacks, use $\text{perturb\_iters} = 10$ and $\alpha = 1.85*(\epsilon/\text{perturb\_iters})$. Comment on the difference between the attacks. Do either of the attacks induce the equivalent of "random guessing" accuracy? If so, which attack and at what ϵ value? (Note: in the code, problem (b) and (c) can be run simultaneously)

(c) (10 pts) *Blackbox Attack* - Using the pre-trained "NetA" as the whitebox model and the pre-trained "NetB" as the blackbox model, measure the ability of adversarial examples generated on the whitebox model to transfer to the blackbox model. Specifically, measure the blackbox classifier's accuracy versus attack epsilon for both the FGSM and PGD attacks. Use the same ϵ values across the range [0, 0.1] and plot the blackbox model's accuracy vs epsilon curve. For the PGD attacks, use $\text{perturb\_iters} = 10$ and $\alpha = 1.85*(\epsilon/\text{perturb\_iters})$. Comment on the difference between the blackbox attacks. Do either of the attacks induce the equivalent of "random guessing" accuracy? If so, which attack and at what ϵ value? (Note: in the code, problem (b) and (c) can be run simultaneously)

(d) (5 pts) Comment on the difference between the attack success rate curves for the whitebox and blackbox attacks. Which is the more powerful attack and why? Does this make sense? Also, consider the epsilon level you found to be the "perceptibility threshold" in Lab 1.b. What is the attack success rate at this level and do you find the result somewhat concerning?

(e) (10 pts) Implement the "Momentum Iterative FGSM" (MI-FGSM) from Algorithm 1 of https://arxiv.org/pdf/1710.06081.pdf in the `attacks.py` file. Note, this should be very similar to your PGD attack (so use copy-paste), with the most noticeable difference being the accumulation of a momentum term throughout the attack iterations. Test the effectiveness of this method as a blackbox attack and compare to the results obtained in part (c) across the same ϵ values, with $\text{perturb\_iters} = 10$ and $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$. Does momentum help? If so, give an intuitive explanation as to why. (Note: MI-FGSM involves a normalization of the gradient, but since these attack functions take a batch of inputs, the norm must be computed separately for each element of the batch)

3 Lab 3: Adversarial Training (40 pts + 10 Bonus)

In this section, you will implement a basic defense called adversarial training (AT). As the name suggests, this involves training the model against adversarial examples. Specifically, we will be using the AT described in https://arxiv.org/pdf/1706.06083.pdf, which formulates the training objective as

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} L(f(x+\delta;\theta),y) \right]$$

Importantly, the inner maximizer specifies that all of the training data should be adversarially perturbed before updating the network parameters. All code for this set of questions will be in the `HWK5_main.ipynb` file. Please include all of your results, figures and observations into your PDF report.

Lab 3 (40 points + 10 Bonus)

(a) (5 pts) Using the given "Model Training" code, adversarially train a "NetA" model using a FGSM attack with $\epsilon = 0.1$, and save the model checkpoint as "netA_advtrain_fgsm0p1.pt". What is the final training accuracy of this model on the clean test data? Is the accuracy less than the standard trained model?

(b) (5 pts) Using the given "Model Training" code, adversarially train a "NetA" model using a PGD attack with $\epsilon = 0.1$, $\text{perturb\_iters} = 4$, $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$, and save the model checkpoint as "netA_advtrain_pgd0p1.pt". What is the final training accuracy of this model on the clean test data? Is the accuracy less than the standard trained model? Are there any noticeable differences in the training convergence between the FGSM-based and PGD-based AT procedures?

(c) (15 pts) For the model adversarially trained with FGSM ("netA_advtrain_fgsm0p1.pt"), compute the accuracy versus attack epsilon curves against both the FGSM and PGD attacks (as whitebox methods only). Use $\epsilon = [0.0, 0.02, 0.04, \ldots, 0.12]$. Is the model robust to both types of attack? If not, explain and analyze why one attack might be better than another. (Note: you can run this code in the "Test Robust Models" cell of the `HWK5_main.ipynb` notebook).

(d) (15 pts) For the model adversarially trained with PGD ("netA_advtrain_pgd0p1.pt"), compute the accuracy versus attack epsilon curves against both the FGSM and PGD attacks (as whitebox methods only). Use $\epsilon = [0.0, 0.02, 0.04, \ldots, 0.12]$, $\text{perturb\_iters} = 10$, $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$. Is the model robust to both types of attack? Explain why or why not and analyze. Can you conclude that one adversarial training method is better than the other? If so, provide an intuitive explanation as to why (this paper may help explain: https://arxiv.org/pdf/2001.03994.pdf). (Note: you can run this code in the "Test Robust Models" cell of the `HWK5_main.ipynb` notebook).

(e) (Bonus 5 pts) Using PGD-based AT, train a few more models with different training ϵ values. Is there a trade-off between clean data accuracy and training ϵ? Is there a trade-off between robustness and training ϵ? What happens when the testing PGD's ϵ is larger than the ϵ used for training? In the report, provide answers to all of these questions along with evidence (e.g., plots and/or tables) to substantiate your claims.

(f) (Bonus 5 pts) Plot the saliency maps for a few samples from the `FashionMNIST` test set as measured on both the standard (non-AT) and PGD-AT models. Do you notice any difference in saliency? What does this difference tell us about the representation that has been learned? (Hint: plotting the gradient w.r.t. the data is often considered a version of saliency, see https://arxiv.org/pdf/1706.03825.pdf)