,

Homework Assignment 2

Due: **2:00pm Thursday, 7 December 2023 on CISPA CMS**

The purpose of this assignment is to get you familiar with black-box attacks, empirical defenses such as adversarial training and certified defenses such as randomized smoothing introduced in Lectures 4-5.

Collaboration Policy: You should do this assignment by yourself and submit your own answers. You may discuss the problems with anyone you want and it is also fine to get help from anyone on problems with LaTeX or Jupyter/Python. You should note in the *Collaborators* box below the people you collaborated with.

Collaborators: TODO: Sneha Chetani (practical collaborator), Subrat Dutta (theory only)

This problem set includes both PDF and Jupyter notebook components. You should complete the answers to the PDF part by writing your answers in `hw2.tex`, and submitting your generated PDF file in CISPA CMS under Submission tab. Similar to the first assignment, the first thing you should do in hw2.tex is setting up your name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your name and your Matr. ID, e.g., `\submitter{Susan Blake (7583916)}`. Before submitting your PDF, also remember to (1) list your collaborators by replacing the TODO in `\collaborators{TODO: replace ...}`, and (2) replace the second line in hw2.tex, `\usepackage{macro}` with `\usepackage[response]{macro}` so the directions do not appear in your final PDF.

Problem 1 (10 pts) Consider the same setting of binary logistic regression as in Problems 1-2 of Homework Assignment 1. Recall from Lecture 4, adversarial training aims to solve the following min-max optimization problem using projected gradient descent (PGD) in the context of logistic regression:

$$\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mu}\left[\max_{\boldsymbol{x}'\in\mathcal{B}_\epsilon(\boldsymbol{x},\ell_\infty)} -\log\sigma(y\cdot\langle\boldsymbol{w},\boldsymbol{x}'\rangle)\right], \tag{1}$$

where μ represents the underlying data distribution. **Write down the pseudocode of PGD-based adversarial training algorithm.** The input of the algorithm should

Algorithm 1 PGD Adversarial Examples

Require: Model $model$, Input X, True label y, Perturbation limit ϵ, Step size α, Number of iterations num_iter, Randomize flag $randomize$
Ensure: Adversarial perturbation δ

$\delta \leftarrow$ zeros_like(X)
criterion $\leftarrow$ CrossEntropyLoss()
for $t = 1$ **to** num_iter **do**
    $pred \leftarrow$ model($X + \delta$)
    $output \leftarrow$ criterion($pred$, y)
    $output$.backward()
    δ.data $\leftarrow \delta$.data $+ \delta$.grad.detach()
    δ.data $\leftarrow$ clamp(δ.data, $-\epsilon, \epsilon$)
    δ.grad.zero_()
end for
return δ

be a set of m training examples $\{(\boldsymbol{x}_i, y_i)\}_{i\in[m]}$ sampled from μ and all the necessary hyperparameters, such as perturbation budget ϵ, attack step size α, number of attack steps S, learning rate η, number of training epochs T, batch size B and etc. The output should be a weight vector $\widehat{\boldsymbol{w}}$ that is supposed to be a good solution to problem (1).

Note: You may want to look into the original paper [MMS+17] that puts forward the algorithm of PGD-based adversarial training for specific algorithmic/implementation details.

Algorithm 2 Adversarial Training/Evaluation Epoch

Require: DataLoader $loader$, Model $model$, Attack method $attack$, Optimizer opt
Ensure: Updated Model $model$

for each batch (i, j) in $loader$ **do**
    $delta \leftarrow$ pgd($model, i, j$, num_iter)
    $pred \leftarrow$ model($i + \delta$)
    $loss \leftarrow$ nn.CrossEntropyLoss()($pred, j$)
    if opt is not None **then**
        opt.zero_grad()
        $loss$.backward()
        opt.step()
    end if
end for
return $model$

Problem 2 (10 pts) Suppose f is a two-class linear classifier with parameters $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, where $f(\boldsymbol{x}) = \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$ for any $\boldsymbol{x} \in \mathbb{R}^d$ and $\text{sgn}(\cdot)$ is the sign function. Suppose g represents the smoothed version of f used for randomized smoothing. Specifically for any $\boldsymbol{x} \in \mathbb{R}^d$, g is defined as:

$$g(\boldsymbol{x}) = \underset{j \in \{-1,+1\}}{\text{argmax}} \; \mathbb{P}_{\boldsymbol{\delta}} \Big[f(\boldsymbol{x} + \boldsymbol{\delta}) = j \Big], \text{ where } \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}),$$

where $\sigma > 0$ is the smoothing parameter. Prove that the following two statements:

1. (5 pts) For any input $\boldsymbol{x} \in \mathbb{R}^d$, $g(\boldsymbol{x}) = f(\boldsymbol{x})$.
2. (5 pts) For any input $\boldsymbol{x} \in \mathbb{R}^d$, the certified radius of g at $\boldsymbol{x}$ with ℓ_2-norm is $R(\boldsymbol{x}) = \frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b|}{\|\boldsymbol{w}\|_2}$. In other words, $g(\boldsymbol{x} + \boldsymbol{\delta})$ remains the same for any $\boldsymbol{\delta} \in \mathbb{R}^d$ with $\|\boldsymbol{\delta}\|_2 \leq R(\boldsymbol{x})$.

Note: The first result implies that the smoothed classifier g is by design identical to the base classifier f, provided that the base classifier f is a two-class linear model, while the second results suggests that the certified radius defined by Theorem 1 in [CRK19] is always "tight" for two-class linear classifiers.

Answers

1. From the concept of randomized smoothing:

$$g(\mathbf{x}) = \arg \max_{j \in \{-1,+1\}} \Pr_{\boldsymbol{\delta}} [\langle w, (\mathbf{x} + \boldsymbol{\delta}) \rangle + b = j]$$

$$= g(\mathbf{x}) = \arg \max_{j \in \{-1,+1\}} \Pr_{\boldsymbol{\delta}} [\langle w, \mathbf{x} \rangle + \langle w, \boldsymbol{\delta} \rangle + b = j]$$

Now, from the definition of randomized smoothing (from slide 53 of PPT 5), the output is the majority vote of the predicted label of many Gaussian-corrupted images. So, when we find these for many images, we check the value of the delta (corruption) in expectation.

To have $f(\mathbf{x}) \neq g(\mathbf{x})$:

$$|\langle w, \mathbf{x} \rangle + b| < |\langle w, \boldsymbol{\delta} \rangle|$$

(as it is done over many images)

$$\mathbb{E}\,|[\langle w, \mathbf{x}\rangle + b]| < \mathbb{E}\,|[\langle w, \boldsymbol{\delta}\rangle]|$$

$$= \mathbb{E}\,|[\langle w, \mathbf{x}\rangle + b]| < |\langle w, \mathbb{E}[\boldsymbol{\delta}]\rangle|$$

Now, as it is Gaussian with mean 0, $\mathbb{E}[\boldsymbol{\delta}] = 0$,

So, $\mathbb{E}\,|[\langle w, \mathbf{x}\rangle + b]| < 0$,

which is a contradiction. Hence, $f(\mathbf{x}) = g(\mathbf{x})$.

2.

Let's consider $f(\mathbf{x}) = \text{sgn}\,(\langle \mathbf{w}, \mathbf{x}\rangle + b)$

So, $f(\mathbf{x}') = f(\mathbf{x} + \boldsymbol{\delta}) = \text{sgn}\,(\langle \mathbf{w}, \mathbf{x} + \boldsymbol{\delta}\rangle + b)$

$$= \text{sgn}\,(\langle \mathbf{w}, \mathbf{x}\rangle + \langle \mathbf{w}, \boldsymbol{\delta}\rangle + b)$$

Now, to make the sign change from f(x) to g(x), the following needs to satisfy:

$$|\langle \mathbf{w}, \mathbf{x}\rangle + b| < |\langle \mathbf{w}, \boldsymbol{\delta}\rangle|$$

Now, from the Cauchy-Schwarz inequality, we know:

$$|\langle \mathbf{w}, \boldsymbol{\delta}\rangle| \leq \|w\|\|\boldsymbol{\delta}\|$$

Applying the Cauchy-Schwarz inequality to the above equation, we get:

$$|\langle \mathbf{w}, \mathbf{x}\rangle + b| \leq \|w\|\|\boldsymbol{\delta}\|$$

$$= \frac{|\langle \mathbf{w}, \mathbf{x}\rangle + b|}{\|\mathbf{w}\|_2} \leq \|\boldsymbol{\delta}\|$$

Which is a contradiction, as $\|\boldsymbol{\delta}\|$ is always bounded by $R(\mathbf{x})$, hence proved.

Implementation Problem. Below is an implementation problem that you need to complete the provided Jupyter notebook. More specifically, you will need to implement both standard training and PGD-based adversarial training using the MNIST handwritten digits dataset. Then, you will evaluate the learned models against different adversarial attacks.

If you haven't used Jupyter notebook before, you can start by installing Jupyter on your computer using this link: https://jupyter.org/install. To run the provided `hw2.ipynb` file (especially for the PGD-based adversarial training part) efficiently, it is necessary to use a GPU instead of CPU. The easiest way is to use the free GPU on Google Collab to run the jupyter notebook. If you happen to have access to more powerful GPUs, you are also suggested to use them for fast computation, but the free GPU provided by Google Collab should be sufficient for you to complete this homework assignment.

Problem 3 (20 pts) Consider the classification task on MNIST and ℓ_∞ perturbations with $\epsilon = 0.1$. Suppose we want to train a CNN model using standard deep learning and a robust CNN model using PGD-based adversarial training. The CNN architecture has 4 convolutional layers and 2 MLP layers. We use a SGD optimizer with learning rate 0.1 for both methods, and both models are trained for 5 epochs.

Your Task: Complete the corresponding functions in the provided Jupyter notebook. Report the test classification errors of both standard-trained and adversarially-trained models in terms of no attack (clean), FGSM attack and PGD attack, respectively, and discuss the results.

Note: The hyperparameters for PGD attack is set as attack step size `alpha=0.02`, number of PGD steps `num_iter=10`. The option `randomize` is for the initialization scheme for PGD-based adversarial training: if `randomize=False`, we use a zero-initialization scheme like FGSM; otherwise, we use a random-initialization scheme. You can choose to use either option of the two initialization schemes.

Problem 4 (bonus, 10 pts) Your task is to develop a function my_attack() that can produce more adversarial examples for MNIST testing examples (i.e., a higher attack success rate) in the Jupyter notebook. Your attack should be valid in the sense that it only produces adversarial perturbations within the ℓ_∞-norm ball with $\epsilon = 0.1$. Also, you need to briefly describe how your attack methodology is developed and what is the attack success rate (ASR) it can achieve.

Note: You may want to implement different attack methods that we learned during class, or varying the hyperparameters for different methods. The higher the attack success rates it can achieve, the higher bonus credits you may receive.

End of Homework Assignment 2 (PDF part)
Don't forget to also complete and submit the Jupyter notebook!

References

[CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[MMS+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.