CPSC 4710/5710: Trustworthy Deep Learning **Fall 2025**

Homework 3

Due TBA

This problem set should be completed individually.

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible.

Submission instructions: You should submit your answers in a PDF file written using LaTeX.

Submitting answers: Prepare answers to your homework in a single PDF file. Make sure that the answer to each sub-question is on a *separate page*. The number of questions should be at the top of each page.

Honor Code: When submitting the assignment, you agree to adhere to the Yale Honor Code. Please read carefully to understand what it entails!

Notations. Unless explicitly stated otherwise, we will adhere to the following conventions.

- A lowercase letter (e.g., n) denotes a number.
- An uppercase letter (e.g., N, S, T) denotes a set and its lowercase (e.g., n, s, t) denotes the set's size.
- A lowercase bold letter (e.g., $\mathbf{x}$) denotes a vector.
- An uppercase bold letter (e.g., $\mathbf{A}$) denotes a matrix.
- A math calligraphic letter denote (e.g., $\mathcal{X}$) denotes a space.

1 Adversarial Defense

In supervised learning, we usually need to choose a classifier f from a set of possible classifiers $\mathcal{F}$ parametrized by θ that best describes data. To measure the good or bad of a classifier under the true data distribution, we often need to define a loss function. The risk of a classifier measures how much, on average, it hurts to use f_θ as the classifier to predict our data. The (true) risk can be defined by:

$$R(f_\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(f_\theta(\mathbf{x}), y)].$$

In practice, the true data distribution is not available. Instead, we can approximate the risk for a given finite dataset $D = \{(\mathbf{x}_i, y_i) \sim \mathcal{D}\}, i = 1, \ldots, m$, which is called *empirical risk*. Picking the classifier f to minimize the empirical risk is known as *empirical risk minimization*, which is defined as follows.

$$\min_\theta \hat{R}\left(f_\theta, D\right) = \min_\theta \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell\left(f_\theta(\mathbf{x}), y\right).$$

To solve the above optimization problem, one common solution for general deep learning models is using a gradient descent algorithm. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point.

In the lectures, we already learned that deep models trained by the vanilla gradient descent algorithm might be susceptible to adversarial attacks. In this section, we learn how to defend against such attacks via robust optimization. Let $\Delta_p = \{\delta : \|\delta\|_p \leq \epsilon\}$ be the set of feasible perturbations. We can define the worst-case adversarial risk as follows

$$R_{\text{adv}}\left(f_\theta, \Delta_p\right) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{\delta\in\Delta_p} \left[\ell(f_\theta(\mathbf{x}+\delta), y)\right].$$

This yields the adversarial training (AT) approach for training robust classifiers:

$$\min_\theta \hat{R}_{\text{adv}}\left(f_\theta, \Delta_p, D\right) = \min_\theta \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \max_{\delta\in\Delta_p} \ell\left(f_\theta(\mathbf{x}+\delta), y\right). \tag{1}$$

Questions. (35pts)

a) Consider a binary classification task recognizing labels $y \in \{0, 1\}$ with a logistic regression model f, *i.e.*,

$$\Pr(Y = y|\mathbf{X} = \mathbf{x}) = f(\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x} + b)^y \cdot [1 - \sigma(\mathbf{w}^\top\mathbf{x} + b)]^{1-y},$$

where σ is the sigmoid function. Let ℓ be the binary cross-entropy loss, and the distance function d be the L2 norm, *i.e.*, $d(\mathbf{x}, \mathbf{x}_0) = \|\mathbf{x} - \mathbf{x}_0\|_2$. Simplify the AT optimization problem (1) by solving the inner max problem. Provide an intuitive comparison between the impact of the term $\epsilon\|\mathbf{w}\|_2$ (should be in your final objective function) in the objective of AT and the objective of the norm-regularized linear regression

$$\min_{\mathbf{w},b} \sum_{(\mathbf{x},y)\in D} \ell(f_{\mathbf{w},b}(\mathbf{x}), y) + \epsilon\|\mathbf{w}\|_2.$$

b) Write the update of the gradient descent algorithm to solve the AT optimization problem in question 1a). Note that you need to compute the gradient analytically.

c) If you implement adversarial training for a neural network, which attack method(s) would you use to solve the inner-max problem? Provide a comparison among attack methods you learned in the lecture (pros and cons of FGSM, Deepfool, PGD, CW, JSMA when integrating with AT).

d) *(Extra credit 5%)* Could you suggest better methods to solve the inner max problem?

2 Robustness Verification

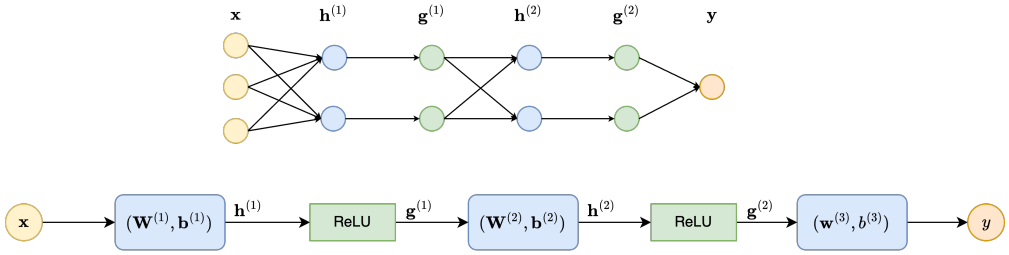

Figure 1: Neural network

Given two-layer neural network with ReLU activation function as described in Figure 1. Specifically, $y = f(\mathbf{x})$ is computed as follows

$$\begin{aligned}
\mathbf{h}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\
\mathbf{g}^{(1)} &= \mathrm{ReLU}(\mathbf{h}^{(1)}) \\
\mathbf{h}^{(2)} &= \mathbf{W}^{(2)}\mathbf{g}^{(1)} + \mathbf{b}^{(2)} \\
\mathbf{g}^{(2)} &= \mathrm{ReLU}(\mathbf{h}^{(2)}) \\
y &= (\mathbf{w}^{(3)})^{\top}\mathbf{g}^{(2)} + b^{(3)},
\end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^3, y \in \mathbb{R}, \mathbf{W}^{(1)} \in \mathbb{R}^{2\times 3}, \mathbf{b}^{(1)} \in \mathbb{R}^2, \mathbf{W}^{(2)} \in \mathbb{R}^{2\times 2}, \mathbf{b}^{(2)} \in \mathbb{R}^2, \mathbf{w}^{(3)} \in \mathbb{R}^2, b^{(3)} \in \mathbb{R}$.

Questions. (35pts)

a) Given two vectors $\mathbf{w}, \mathbf{h} \in \mathbb{R}^d$, where $\mathbf{h}$ is bounded by $\mathbf{l} \le \mathbf{h} \le \mathbf{u}$ (element-wise: $l_i \le h_i \le u_i$), show that we can find a diagonal matrix $\mathbf{D}$ and a vector $\mathbf{c}$ such that

$$\mathbf{w}^{\top}\,\mathrm{ReLU}(\mathbf{h}) \ge \mathbf{w}^{\top}(\mathbf{D}\mathbf{h} + \mathbf{c}),$$

where ReLU is an element-wise activation function and $\mathbf{D}$ is a diagonal matrix and $\mathbf{c}$ is a vector.

(Hint: Split the domain into three cases ($u_i < 0$, $l_i \le 0 \le u_i$, and $0 < l_i$) and analyze the lower bound of ReLU function for each case. Choose the tightest lower bound of the ReLU *function for each case. You may need to add some free variables to the lower bound.)*

b) Given $\mathbf{x} \in \mathcal{S} = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_0\|_\infty \le \epsilon\}$ where $\epsilon = 1.0$ and

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}; \mathbf{W}^{(1)} = \begin{bmatrix} 2 & -1 & 1 \\ 1 & 1 & -2 \end{bmatrix}; \mathbf{b}^{(1)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 0 & -1 \\ -2 & 1 \end{bmatrix}; \mathbf{b}^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \mathbf{w}^{(3)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}; b^{(3)} = -1.$$

Compute the lower and upper bounds for $\mathbf{h}^{(1)}, \mathbf{g}^{(1)}, \mathbf{h}^{(2)}, \mathbf{g}^{(2)}$, and y by forward-passing the domain $\mathcal{S}$ of $\mathbf{x}$ through the neural network.

c) Find a better lower bound for $y = f(\mathbf{x})$ obtained in 2b) by propagating the CROWN lower bound backward from the output to the input (any lower bound better than the naive lower bound found in (2b) would be sufficient).

(Hint: For linear layers, propagate the weight $\mathbf{W}^{(l)}$. For ReLU layers, determine $\mathbf{D}$ and $\mathbf{c}$ using the result in 2a. Note that there are two ways you can improve the CROWN lower bound, one is to choose a suitable free variable for $\mathbf{D}$ in the CROWN lower bound; and the other is to improve the lower and upper bound $(\mathbf{l}, \mathbf{u})$ for the input of the ReLU layer.)