EECS 127 Project: Adversarial Machine Learning

Justin Wu, Siddharth Nath, Stephen Su

April 29, 2024

Abstract

In this project, we will be exploring how to make neural net classifiers robust by framing them as a non-convex optimization problem, then using the techniques we learned in this class to prove their robustness. In particular, we will follow a technique developed by Wong and Kolter.

1 Introduction

Deep neural networks dominate fields like computer vision and natural language processing but are vulnerable to adversarial attacks, where small input perturbations can cause significant errors. This project employs Lagrangian duality to defend against such attacks by certifying the robustness of classifiers. Through rigorous optimization, it assesses the worst-case scenario to ensure resilience and facilitates the re-training of more secure classifiers, enhancing the reliability of deep learning systems in safety-critical applications.

2 Related Work

2.1 Towards Evaluating the Robustness of Neural Networks by Carlina and Wagner

Despite the ability of neural networks to provide high-quality results for many machine-learning tasks, neural networks are not perfect. These networks are vulnerable to adversarial attacks, ones that are designed to exploit vulnerabilities in ML systems by deceiving the model to classify inputs incorrectly. Since ML is heavily integrated into our daily lives, including various applications controlling our health and well-being, the threat of adversarial attacks is significant!

The authors of this paper present three new attacks that they hope will serve as an industry benchmark for others attempting to create robust neural networks that resist adversarial examples. This paper is a significant breakthrough in Adversarial Machine Learning since it can penetrate networks strengthened by defensive distillation, a technique initially known for improving model robustness against adversarial examples.

Defensive Distillation relies on the concept of distillation which is implemented in four steps:

1. First, train a teacher model on the training set in a standard manner
2. Then, using the teacher model, you label each instance of the training set with its soft labels (confidence for each class) instead of its hard label (the class with the highest confidence). These soft labels contain richer information about the relationships between different classes as perceived by the model.
3. Using a large distillation temperature to force the distilled model to become more confident in its predictions, train the distilled network on the computed soft labels

4. Finally, reset the temperature to 1 when using the distilled network to classify new inputs

The intuition behind why Defensive Distillation works lies in the fact that increasing the temperature pushes the softmax function to a more uniform distribution. Therefore, training the distilled network on soft labels generated at a high temperature produces smoother probability distributions over classes. This smoothness makes it harder for small perturbations (like those used in adversarial attacks) to push a prediction across a decision boundary.

To describe previous attack algorithms, consider L-BFGS. This algorithm is used to generate adversarial examples using the following optimization problem:

$$\min \|x - x'\|_2^2$$
$$\text{such that } C(x') = l, x' \in [0,1]^n$$

The three new attack algorithms presented in this paper are for the L_0, L_2, and L_∞ distance metrics. The attacks are constructed according to the norm constraints applied to the perturbations that define different ways of measuring the magnitude of changes made to the original input to generate the adversarial example.

The L2 attack uses gradient descent to converge to adversarial points based on minimizing the following objective function. Given x and a target class t such:

$$\min \left\| \frac{1}{2}(\tanh w + 1) - x \right\|_2^2 + c * f(\frac{1}{2}(\tanh w + 1))$$

The idea behind this attack is to choose a target class t given input x such that $t \neq C^*(x)$

Since gradient descent can get stuck at a local minimum, the authors decided to pick many random starting points close to the original image. This helps reduce the possibility of gradient descent getting such at an un-optimal local minimum.

2.2 Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope by Wong and Kolter

This paper describes a method for training provably robust classifiers that are guaranteed to be robust against any norm-bounded adversarial perturbations. Additionally, the authors claim that their method can detect any previously unseen adversarial examples with zero false negatives.

To accomplish this, the presented method relies on constructing a convex outer bound on the "adversarial polytope" which is the set of all final-layer activations that can be achieved by applying a norm-bounded perturbation to the input. In other words, the adversarial polytope represents the set of activations that any input perturbation can reach.

The adversarial polytope is the set of all activations attained by perturbing x by some Δ with l_∞ norm bounded by ϵ. This is defined as the set $Z_\epsilon(x) = f_\theta(x + \Delta) : \|\Delta\|_\infty \leq \epsilon$

However, since the adversarial polytope is not guaranteed to be convex, it can be difficult to compute and optimize over. To make this problem more computationally feasible, the authors introduce a convex relaxation of the polytope which is essential because the convexity allows for the use of convex optimization techniques.

This paper considers a k-layer feedforward ReLU-based neural network. The paper describes that the starting point of the convex outer bound is a linear relaxation of the ReLU activations which is done by replacing the ReLU inequalities with their upper convex envelopes.

Using this convex outer bound, the method in the paper can efficiently compute and optimize the worst-case loss to learn classifiers that are provably robust to the perturbations defined by the adversarial polytope.

The method in the paper is derived in three steps:

1. Define the adversarial polytope for deep ReLU networks and present the convex outer bound
2. Optimize over the convex outer bound using the dual problem
3. Using the dual approach, compute the necessary upper and lower activation bounds

The primal optimization problem that is used to generate adversarial examples aims to find the worst-case perturbation of the input. Given a sample x with known label y^*, we can find the point in $Z_\epsilon(x)$ that minimizes this class and maximizes some alternative target y^{targ} by optimizing:

$$\min_{\hat{z_k}} (\hat{z_k})_{y^*} - (\hat{z_k})_{y^{\text{targ}}} = c^\top \hat{z_k}$$

$$\text{subject to } \hat{z_k} \in Z_\epsilon(x)$$

Although the above optimization problem is an LP, solving an LP via traditional methods for each target class is not the most feasible. Specifically, the LP above has the same number of variables as activation layers in the network. Therefore, the authors approach the dual of the above optimization problem to obtain a robust neural network. Since we know that any feasible dual solution provides a guaranteed lower bound on the solution of the primal consequently means that we can use the dual to get a provable bound on the adversarial error.

The relationship between Wong and Kolter's Paper and Carlini and Wagner's Paper is that Wong and Kolter's work builds on adversarial model robustness by providing a methodology for training neural networks that are provably robust against adversarial attacks. Their approach uses convex outer approximations of adversarial polytopes to accomplish this.

On the other hand, Carlini and Wagner's paper introduces effective adversarial attack methods that challenges many previous methods that were proposed to defend against adversarial examples. This paper demonstrates that many of the existing defenses were not as robust as believed. Their work is often used as a benchmark to test the robustness of defense mechanisms.

The relationship between these two papers is that they both build on top of the concerns that surround adversarial machine learning. Wong and Kolter's paper can be viewed as a direct response to the need for more robust defenses highlighted by the effectiveness of Carlini and Wagner's attack techniques. The work in this paper uses the insights gained from previous attacks to develop a more robust and provable defense mechanism against adversarial attacks.

2.3 Theoretically Principled Trade-off between Robustness and Accuracy

The paper theoretically categorizes the trade-off between robustness and accuracy of a classifier. It defines prediction error for adversarial examples as the sum of natural (classification) error and "boundary error" - how likely inputs are to be close to the decision boundary. The objective is therefore to minimize this robust error. To address this objective function, the paper purposes a defense method called TRADES that balances natural error with a regularization term to encourage decision boundary to be far from the data points. As a result, TRADES achieves state-of-the-art robustness on benchmarks.

This paper introduces the concept of boundary error into evaluating classifier robustness. Unlike the previous two papers, which discuss the attack and defense mechanisms on designing a robust classifiers, this paper examines the theoretical nature of robustness and its effect on other aspects of classifier performance. Additionally, the TRADES algorithm discusses the effect of regularization on decision boundaries, which is a concept that was omitted from previous papers.

This paper provides an understanding of the tensions in making models both accurate and robust, quantifies this tension by proving a tight bound on robust error, and proposes a state-of-the-art method for training robust models while taking accuracy into consideration.

Some open questions are:

- The boundary error term is closely related to the notion of margin in classification. Can the results in this paper be connected to or combined with existing margin-based generalization bounds in learning theory?
- The theoretical analysis in the paper focuses on binary classification. How can these results, especially the decomposition of robust error into natural error and boundary error, be extended to multi-class problems?
- The paper demonstrates the inherent trade-off between robustness and accuracy in image classification. Does this trade-off exist in other domains, such as natural language processing, speech recognition, or reinforcement learning?

3 Problems

Please complete the guided problem set in the project assignment. Feel free to add more sections/subsections as necessary.

3.2 3.2.1 Interpretation of FGSM

Solution: We start by substituting $x' = x + \epsilon v$. Our formulation becomes

$$\arg\max_v L(f_\theta(x), y_{true}) + (\nabla_x L(f_\theta(x), y_{true}))^\top (x + \epsilon v)$$

$$\text{s.t.} ||v||_\infty \leq 1$$

Since many of the terms in our argmax function do not depend on v, we can remove some of the nondependent terms to get an equivalent problem

$$\arg\max_v \nabla_x L(f_\theta(x), y_{true})^\top v$$

$$\text{s.t.} ||v||_\infty \leq 1$$

By the dual of the infinity norm, the solution to the maximization of this problem is $||\nabla_x L(f_\theta(x), y_{true})||_1$, the ℓ_1 norm of the gradient. To achieve this, we have to recognize that the ℓ_1 norm is the absolute value of each term of the gradient. Thus, the argmax of this problem will be

$$v = \text{sgn}(\nabla_x L(f_\theta(x), y_{true}))$$

Plugging this back into x', we get that $x' = x + \epsilon \cdot \text{sgn}(\nabla_x L(f_\theta(x), y_{true}))$ is the argmax to our original problem.

When the constraint becomes $||x - x'||_2 \leq \epsilon$, the solution becomes different. If we use the same subsitution $x' = x + \epsilon v$, we get the following problem:

$$\arg\max_v L(f_\theta(x), y_{true}) + (\nabla_x L(f_\theta(x), y_{true}))^\top (x + \epsilon v)$$

$$\text{s.t.} ||v||_2 \leq 1$$

Using the same techniques to remove nondependent terms, we simplify our problem into

$$\arg\max_v \nabla_x L(f_\theta(x), y_{true})^\top v$$

$$\text{s.t.}||v||_2 \leq 1$$

By the Cauchy-Schwartz inequality, $\nabla_x L(f_\theta(x), y_{true})^\top v \leq ||\nabla_x L(f_\theta(x), y_{true})||_2 ||v||_2 \leq ||\nabla_x L(f_\theta(x), y_{true})||_2$. To achieve our upper bound, we set v to be in the same direction as the gradient. However, since the magnitude of v is bounded by 1, we need to set v to be a unit vector in the same direction as the gradient. Thus, we get that

$$v^* = \frac{\nabla_x L(f_\theta(x), y_{true})}{||\nabla_x L(f_\theta(x), y_{true})||_2}$$

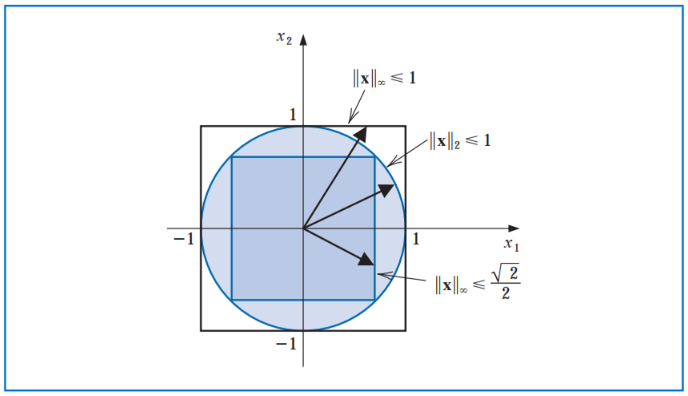
Plugging this back into our original x', we get that

$$x' = x + \epsilon \cdot \frac{\nabla_x L(f_\theta(x), y_{true})}{||\nabla_x L(f_\theta(x), y_{true})||_2}$$

The solution under the ℓ_2 norm is different from the solution under the ℓ_∞ norm, because the solution set for ℓ_2 is different from the solution set for ℓ_∞. The solution set for ℓ_2 is contained with in a ball, and the optimal solution would be along the radius of the ball. The solution set for the ℓ_∞ norm is a box. Solutions here will be on one of the edges of the box or in one of the corners of the box. These two constraints produce different solution sets, which then in turn produce different solutions.

Figure 7.3 illustrates this result when $n = 2$.

Figure 7.3


x_2
$\|\mathbf{x}\|_\infty \leqslant 1$
$\|\mathbf{x}\|_2 \leqslant 1$
1
-1
1
x_1
$\|\mathbf{x}\|_\infty \leqslant \frac{\sqrt{2}}{2}$
-1

3.3 Convex relaxation of the adversarial optimization

Solution: To modify the problem such that a negative objective value implies that our classifier outputs $\vec{y}_{targ}$, we will consider a new optimization problem:

$$\begin{aligned} \min_{\vec{z}} \max_{i \neq targ} \; & z_{3i} - z_{3targ} \\ \text{s.t.} \; & ||\vec{z}_1 - \vec{x}||_\infty \leq \epsilon \\ & \vec{\hat{z}}_2 = W_1 \vec{z}_1 + \vec{b}_1 \\ & \vec{z}_2 = \text{ReLU}(\vec{\hat{z}}_2) \\ & \vec{\hat{z}}_3 = W_2 \vec{z}_2 + \vec{b}_2 \end{aligned}$$

Our objective function becomes a maximization over the difference between each output and z_{3targ}. In the case where the maximum difference is negative, our objective function implies that $z_{3targ} > z_i$ for all $i \neq targ$. As a result, z_{3targ} has to be the maximum over all z_i's. Thus, the classifier will choose $\vec{y}_{targ}$ as the output when the objective function is negative.

3.4 Fenchel Conjugates

Solution:

(a) Suppose f:

i. Case 1: $y < -1$:

$$\text{if } x > 0 \to yx - x = x(y-1)$$

$$\text{if } x < 0 \to yx - |x| = yx - (-x) = yx + x = x(y+1)$$

$$\text{As } x \text{ approaches } \infty,\ yx - |x| \to \infty \implies f^*(y) = \infty$$

ii. Case 2: $y = -1$

$$yx - |x| = (-1)x - |x| = -x - |x|$$

$$\text{if } x > 0 \to -x - |x| = -2x$$

$$\text{if } x < 0 \to -x - |x| = 0$$

$$\implies f^*(y) = \infty$$

iii. Case 3: $-1 < y < 1$

$$\text{if } x > 0 \to yx - |x| = yx - x = x(y-1) < 0$$

$$\text{if } x < 0 \to yx - |x| = yx - (-x) = x(y+1) < 0$$

$$\text{if } x = 0 \to yx - |x| = 0$$

$$\implies f^*(y) = 0$$

iv. Case 4: $y = 1$

$$yx - |x| = (1)x - |x| = x - |x|$$

$$\text{if } x > 0 \to x - |x| = 0$$

$$\text{if } x < 0 \to x - |x| = x - (-x) = 2x < 0$$

$$\implies f^*(y) = 0$$

v. Case 5: $y > 1$

$$\text{if } x > 0 \to yx - |x| = yx - x = x(y-1)$$

$$\text{if } x < 0 \to yx - |x| = yx - (-x) = x(y+1)$$

$$\text{As } x \text{ approaches } \infty,\ yx - |x| \to \infty \implies f^*(y) = \infty$$

From these results, we get that

$$f^*(y) = \begin{cases} 0 & \text{when } |y| \leq 1 \\ \infty & \text{otherwise} \end{cases}$$

(b)

$$f^*(y) = \sup_{\vec{x}}\{\vec{y}^T\vec{x} - \|\vec{x}\|_1 \,|\, \vec{x} \in \mathbb{R}^n\}$$

We can first rewrite $\|\vec{x}\|_1$ as $\sum_{i=1}^n |x_i|$ and the dot product $\vec{y}^\top\vec{x}$ as $\sum_{i=1}^n y_i x_i$

$$\implies f^*(y) = \sup_{\vec{x}}\{\sum_{i=1}^n y_i x_i - \sum_{i=1}^n |x_i| \,|\, \vec{x} \in \mathbb{R}^n\}$$

Looking at the element-wise expression, we see that:

$$\sup_{x_i}\{y_i x_i - |x_i| \,|\, x_i \in \mathbb{R}\} = \begin{cases} 0 & \text{when } |y_i| \leq 1 \\ \infty & \text{otherwise} \end{cases} \qquad \text{from (a)}$$

$$\implies f^*(y) = \begin{cases} 0 & \text{when } |y| \leq 1 \forall\ i \\ \infty & \text{otherwise} \end{cases}$$

$$\implies f^*(y) = \begin{cases} 0 & \text{when } \|\vec{y}\|_\infty \leq 1 \\ \infty & \text{otherwise} \end{cases}$$

3.5 Using Lagrangian Duality

Solution:

(a) Incorporate the constraints $\|\vec{z_1} = \vec{x}\|_\infty \leq \epsilon$ and $(z_{2j}, \hat{z}_{2j}) \in z_j\ \forall\ j \in \{1, ..., n_2\}$ into the objective function:

(i) $\|\vec{z_1} = \vec{x}\|_\infty \leq \epsilon \to \vec{z_1}$ must be in the l_∞ ball $\implies \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1})$

(ii) $(z_{2j}, \hat{z}_{2j}) \in z_j\ \forall\ j \in \{1, ..., n_2\} \implies \mathbf{1}_{z_j}(z_{2j}, \hat{z}_{2j})\ \forall\ j$

$$\implies p^*(\vec{x}, \vec{c}) = \min_{\vec{x}} \vec{c}^T\vec{\hat{z_3}} + \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) + \sum_{i=1}^{n_2} \mathbf{1}_{z_j}(z_{2j}, \hat{z}_{2j})$$

$$\text{such that } \vec{\hat{z_2}} = W_1\vec{z_1} + \vec{b_1} \text{ and } \vec{\hat{z_3}} = W_2\vec{z_2} + \vec{b_2}$$

(b) Let $\vec{\nu_3}$ be the dual variable for $\vec{\hat{z_3}} = W_2\vec{z_2} + \vec{b_2}$ and let $\vec{\nu_2}$ be the dual variable for $\vec{\hat{z_2}} = W_1\vec{z_1} + \vec{b_1}$

$$\implies \begin{cases} \vec{\hat{z_3}} - W_2\vec{z_2} - \vec{b_2} = 0 \\ \vec{\hat{z_2}} - W_1\vec{z_1} - \vec{b_1} = 0 \end{cases}$$

Now, we can write the Lagrangian for the problem from part (a) as:

$$L(x, \nu) = \vec{c}^\top\vec{\hat{z_3}} + \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) + \sum_{i=1}^{n_2} \mathbf{1}_{z_j}(z_{2j}, \hat{z_{2j}}) + \vec{\nu_3}^\top(\vec{\hat{z_3}} - W_2\vec{z_2} - \vec{b_2}) + \vec{\nu_2}^\top(\vec{\hat{z_2}} - W_1\vec{z_1} - \vec{b_1})$$

$$= \vec{c}^\top\vec{\hat{z_3}} + \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) + \sum_{i=1}^{n_2} \mathbf{1}_{z_j}(z_{2j}, \hat{z_{2j}}) + \vec{\nu_3}^\top\vec{\hat{z_3}} - \vec{\nu_3}^\top W_2\vec{z_2} - \vec{\nu_3}^\top\vec{b_2} + \vec{\nu_2}^\top\vec{\hat{z_2}} - \vec{\nu_2}^\top W_1\vec{z_1} - \vec{\nu_2}^\top\vec{b_1}$$

Simplifying further, we can write $\vec{\nu_2}^\top\vec{b_1} - \vec{\nu_3}^\top\vec{b_2}$ as $\sum_{i=1}^2 \vec{\nu_{i+1}}^\top\vec{b_i}$

$$= \vec{c}^\top\vec{\hat{z_3}} + \vec{\nu_3}^\top\vec{\hat{z_3}} + \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1\vec{z_1} + (\sum_{i=1}^{n_2} \mathbf{1}_{z_j}(z_{2j} - \hat{z_{2j}}) - \vec{\nu_3}^\top W_2\vec{z_2} + \vec{\nu_2}^\top\vec{\hat{z_2}}) - \sum_{i=1}^{2} \vec{\nu_{i+1}}^\top\vec{b_i}$$

(c) $g(\vec{\nu_2}, \vec{\nu_3}) = \min_{\vec{z}} L(\vec{z}, \vec{\nu})$

$$= \min_{\vec{z}} \vec{c}^\top \vec{\hat{z_3}} + \vec{\nu_3}^\top \vec{\hat{z_3}} + \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1 \vec{z_1} + (\sum_{i=1}^{n_2} \mathbf{1}_{z_j}(z_{2j} - \hat{z_{2j}}) - \vec{\nu_3}^\top W_2 \vec{z_2} + \vec{\nu_2}^\top \vec{\hat{z_2}}) - \sum_{i=1}^{2} \vec{\nu_{i+1}}^\top \vec{b_i}$$

$$= \min_{\vec{\hat{z_3}}}[(\vec{c}^\top \vec{\hat{z_3}} + \vec{\nu_3}^\top \vec{\hat{z_3}})] + \min_{\vec{z_1}}(\mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1 \vec{z_1}) + (\sum_{j=1}^{n_2} \min_{z_{2j}, \hat{z_{2j}}} (\mathbf{1}_{z_j}(z_{2j} - \hat{z_{2j}}) - \vec{\nu_3}^\top (W_2)_j z_{2j} + \nu_{2j}^\top z_z \hat{2}j) - \sum_{i=1}^{2} \vec{\nu_{i+1}}^\top \vec{b_i}$$

$$= \min_{\vec{\hat{z_3}}}[(\vec{c} + \vec{\nu_3})^\top \vec{\hat{z_3}}] + \min_{\vec{z_1}}(\mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1 \vec{z_1}) + (\sum_{j=1}^{n_2} \min_{z_{2j}, \hat{z_{2j}}} (\mathbf{1}_{z_j}(z_{2j} - \hat{z_{2j}}) - \vec{\nu_3}^\top (W_2)_j z_{2j} + \nu_{2j}^\top z_z \hat{2}j) - \sum_{i=1}^{2} \vec{\nu_{i+1}}^\top \vec{b_i}$$

(d) The Lagrangian dual in this case is:

$$\max_{\vec{v}}(\min_{\vec{\hat{z_3}}}[(\vec{c} + \vec{\nu_3})^\top \vec{\hat{z_3}}] + \min_{\vec{z_1}}[\mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1 \vec{z_1}] + \sum_{j=1}^{n_2} \min_{z_2 j, \widehat{z}_{2j}} [\mathbf{1}_{\mathcal{Z}_|}(z_2 j, \widehat{z}_{2j}) - \vec{\nu_3}^\top (W_2)_j z_{2j} + \nu_{2j} \widehat{z}_{2j}] + \sum_{i=1}^{2} \vec{\nu}_{i+1}^\top \vec{b}_i)$$

Looking at the term

$$\min_{\vec{\hat{z_3}}}[(\vec{c} + \vec{\nu_3})^\top \vec{\hat{z_3}}]$$

this will take on the value of $-\infty$ if $\vec{c} + \vec{\nu_3} \neq 0$, since we can drive $\vec{\hat{z_3}}$ all the way down to $-infty$ to minimize the value. Otherwise, when $\vec{c} + \vec{\nu_3} = 0$, the whole term equals 0. Since we want to maximize this term, we want to ensure that the term is 0, preferable over $-\infty$. Thus, we can add a constraint such that $\vec{\nu_3} = -\vec{c}$

With the term

$$\min_{\vec{z_1}}[\mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1}) - \vec{\nu_2}^\top W_1 \vec{z_1}]$$

we can turn this term into an equivalent maximization:

$$-\max_{\vec{z_1}}[\vec{\nu_2}^\top W_1 \vec{z_1} - \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1})]$$

Because any maximum is also a supremum, we can turn this into

$$-\sup_{\vec{z_1}}[\vec{\nu_2}^\top W_1 \vec{z_1} - \mathbf{1}_{B_\epsilon(\vec{x})}(\vec{z_1})]$$

By definition of the Fenchel conjugate, this becomes

$$-\mathbf{1}^*_{B_\epsilon(\vec{x})}(W_1^\top \vec{\nu_2})$$

Next, with the term

$$\sum_{j=1}^{n_2} \min_{z_2 j, \widehat{z}_{2j}} [\mathbf{1}_{\mathcal{Z}_j}(z_2 j, \widehat{z}_{2j}) - \vec{\nu_3}^\top (W_2)_j z_{2j} + \nu_{2j} \widehat{z}_{2j}]$$

we can use the same trick again to turn this into a maximization, and again turn the term into a supremum. Applying this would get

$$\sum_{j=1}^{n_2} - \sup_{z_2 j, \widehat{z}_{2j}} [\vec{\nu_3}^\top (W_2)_j z_{2j} - \nu_{2j} \widehat{z}_{2j} - \mathbf{1}_{\mathcal{Z}_j}(z_2 j, \widehat{z}_{2j})]$$

Once again, by the definition of the Fenchel conjugate, this becomes

$$\sum_{j=1}^{n_2} -\mathbf{1}^*_{\mathcal{Z}_j}(\vec{\nu_3}^\top (W_2)_j, -\nu_{2j})$$

Putting each term back into the original objective function, we get our dual optimization problem

$$\max_{\vec{v}}(-\mathbf{1}^*_{B_\epsilon(\vec{x})}(W_1^\top \vec{\nu_2}) + \sum_{j=1}^{n_2} -\mathbf{1}^*_{\mathcal{Z}_j}(\vec{\nu_3}^\top (W_2)_j, -\nu_{2j}) + \sum_{i=1}^{2} \vec{\nu}_{i+1}^\top \vec{b}_i)$$

$$\text{s.t. } \vec{\nu_3} = -\vec{c}$$

3.6 Finding the Fenchel Conjugates

Solution:

based on prior definition of the fenchel conjugate and the ϵ-ball

$$\mathbf{1}^*_{B_\epsilon(\vec{x})}(\vec{\nu}) = \sup_{\vec{y}\in B_\epsilon(\vec{x})} \vec{\nu}^\top \vec{y}$$

given $\|y-x\| \leq \epsilon$, $\|y\| \leq x+\epsilon$, rearranging the equation

$$\mathbf{1}^*_{B_\epsilon(\vec{x})}(\vec{\nu}) = \vec{\nu}^\top \vec{x} + \sup_{\|\vec{y}-\vec{x}\|\leq\epsilon} \vec{\nu}^\top (\vec{y}-\vec{x})$$

let $z = y - x$, $\vec{\nu}^\top \vec{z}$ maximized when sign of z_i matches with ν_i and when the magnitude is ϵ

$$\vec{\nu}^\top \vec{z} = \sum_i \nu_i z_i = \sum_i \nu_i \left(\epsilon \cdot \text{sign}\left(\nu_i\right)\right) = \epsilon \sum_i |\nu_i| = \epsilon \|\nu\|_1$$

putting together we get the desired result

3.7 The Dual Network

Solution:

(a) Denote equation 28 as $A+B+C$, where

$$A = -\mathbf{1}^*_{B_\epsilon(\vec{x})}(\vec{\hat{\nu}}_1)$$

$$B = \sum_{j=1}^{n_2} -\mathbf{1}^*_{\mathcal{Z}_j}\left(\hat{\nu}_{2j}, -\nu_{2j}\right)$$

$$C = -\sum_{i=1}^{2} \vec{\nu}_{i+1}^\top \vec{b}_i$$

$A = -\overrightarrow{\hat{\nu}}_1^\top \vec{x} - \epsilon \left\|\overrightarrow{\hat{\nu}}_1\right\|_1$ from part 3.6

B has been incorporated into the three conditions for v_{2j}

C only applies to $j \in S$ and will be zero otherwise

Putting A, B, C together, we get equation 33

(b)

the relu function by itself is convex, but how it contributes to the overall convexity of the optimization is dependent on if its used in constraint or convexity.

the terms in the dual problem are either affine or relu functions. the relu function is also convex. therefore the entire problem is convex.

In the unrelaxed primal problem, the feasible region formed by the relu function is non convex since it has a kink at the origin. visually, there's no line segment that can connect the points on either side of the kink without being outside the feasible region.

3.8 Finding the bounds

Solution: Fro the constraint of $||\vec{z_1} - \vec{x}||_\infty \leq \epsilon$, we get that $\max_i |\vec{z_1}_i - \vec{x}_i| \leq \epsilon$. This implies that $|\vec{z_1}_i - \vec{x}_i| \leq \epsilon$ for all i. We can rewrite this inequality to remove the absolute value, which then becomes $-\epsilon \leq \vec{z_1}_i - \vec{x}_i \leq \epsilon$. We can move the $\vec{x}_i$ term to get that

$$\vec{x}_i - \epsilon \leq \vec{z_1}_i \leq \vec{x}_i + \epsilon$$

We define $\hat{\vec{z_2}} = W_1\vec{z_1} + \vec{b_1}$ by the second layer of the neural network. We can define $\hat{\vec{z_2}}$ element-wise to get $\hat{\vec{z_{2i}}} = \vec{w_i}^\top \vec{z_1} + \vec{b_{1i}}$, where $\vec{w_i}^\top$ is the i^{th} row of W. Since $\vec{z_1}_i$ is upper bounded by $\vec{x}_i + \epsilon$, we can replace $\vec{z_1}$ with $\vec{x} + \epsilon\vec{1}$, where $\vec{1}$ is a vector of all ones. Then,

$$\hat{\vec{z_{2i}}} \leq \vec{w_i}^\top(\vec{x} + \epsilon\vec{1}) + \vec{b_{1i}} = \vec{w_i}^\top\vec{x} + \vec{w_i}^\top\epsilon\vec{1} + \vec{b_{1i}}$$

However, looking at the term $\vec{w_i}^\top\epsilon\vec{1}$, this is equivalent to $\epsilon\sum_j \vec{w_{ij}}$. We can upper bound this to be $\epsilon\sum_j \vec{w_{ij}} \leq \epsilon\sum_j |\vec{w_{ij}}| = \epsilon||\vec{w_i}||_1$. Combining this inequality with our last inequality, we get that

$$\hat{\vec{z_{2i}}} \leq \vec{w_i}^\top\vec{x} + \vec{b_{1i}} + \epsilon||\vec{w_i}||_1$$

We can combine this inequality into one vector to say

$$\hat{\vec{z_2}} \leq W\vec{x} + \vec{b_1} + \epsilon||W_1||_{:1} = \vec{u}$$

Similarly, the lower bound $\vec{l}$ can be calculated as well. We know that

$$\hat{\vec{z_{2i}}} \geq \vec{w_i}^\top(\vec{x} - \epsilon\vec{1}) + \vec{b_{1i}} = \vec{w_i}^\top\vec{x} - \vec{w_i}^\top\epsilon\vec{1} + \vec{b_{1i}}$$

by the lower bound of $\vec{z_1}_i$. The term $\vec{w_i}^\top\epsilon\vec{1} = -\epsilon\sum_j \vec{w_{ij}}$ is lower bounded by $-\epsilon\sum_j |\vec{w_{ij}}|$, since we stated before that $\epsilon\sum_j \vec{w_{ij}} \leq \epsilon\sum_j |\vec{w_{ij}}|$. Also, we can rewrite $-\epsilon\sum_j |\vec{w_{ij}}|$ to be $-\epsilon||\vec{w_i}||_1$. Thus, our overall lower bound becomes

$$\hat{\vec{z_{2i}}} \geq \vec{w_i}^\top\vec{x} + \vec{b_{1i}} - \epsilon\sum_j |\vec{w_{ij}}| = \vec{w_i}^\top\vec{x} + \vec{b_{1i}} - \epsilon||\vec{w_i}||_1$$

Since this is only for one element of $\hat{\vec{z_{2i}}}$, we can take into account every element into one inequality. As a result, we get that

$$\hat{\vec{z_2}} \geq W\vec{x} + \vec{b_1} - \epsilon||W_1||_{:1} = \vec{l}$$

3.9 A certificate for robustness

Solution: N/A

3.10 Training a robust classifier

Solution:

(a) Showing that the cross-entropy loss is monotonic and translation-invariant:

i. Monotonic Proof:

Let $p_i = \dfrac{e^{y_i}}{\sum_{j=1}^m e^{y_j}}$

$$\frac{\partial p_i}{\partial y_i} = \frac{e^{y_i}(\sum_{j=1}^m e^{y_j}) - e^{y_i}(e^{y_i})}{(\sum_{j=i}^m e^{y_j})^2} = \frac{e^{y_i}((\sum_{j=1}^m e^{y_j}) - e^{y_i})}{(\sum_{j=i}^m e^{y_j})^2} = \frac{e^{y_i}}{\sum_{j=i}^m e^{y_j}} * \frac{(\sum_{j=i}^m e^{y_j}) - e^{y_i}}{\sum_{j=i}^m e^{y_j}}$$

$$\implies p_i(1-p_i)$$

For this proof we want to show if $y'_i \geq y_i \; \forall$ incorrect classes $i : L(\vec{y}, \vec{y_{\text{true}}}) \leq L(\vec{y}' vec y_{\text{true}})$
We do this by taking the derivative of the loss function:

$$\frac{\partial L}{\partial y_i} = -\sum_{k=1}^m (y_{\text{true}})_k * \frac{\partial(\log p_k)}{\partial y_i}$$

For $k \neq i \implies \dfrac{\partial(\log p_k)}{\partial y_i} = \dfrac{-1}{p_k}\dfrac{\partial p_k}{\partial y_i}$
For $k = i \implies \dfrac{\partial(\log p_i)}{\partial y_i} = \dfrac{1}{p_i}\dfrac{\partial p_i}{\partial y_i}$
Plugging these derivatives back into the loss function, we get:

$$\frac{\partial L}{\partial y_i} = p_i * (\sum_{k=1}^m (y_{\text{true}})_k) - (y_{\text{true}})_i$$

Looking at the sign of the derivative, we see that increasing p_i (which is achieved with $y'_i \geq y_i$) will keep the derivative positive. Since the sign is positive, the loss will increase.

ii. Translation-Invariant Proof:

$$L(\vec{y} - a\mathbf{1}, \vec{y_{\text{true}}}) = -\sum_{i=1}^m (y_{\text{true}})_i \log(\frac{e^{(y_i - a)}}{\sum_{j=1}^m e^{(y_j - a)}})$$

We can simplify the expression involving the $log()$ as:

$$\log \frac{e^{y_i}e^{-a}}{\sum_{j=1}^m (e^{y_j}e^{-a})} = \log \frac{e^{-a}}{e^{-a}} * \frac{e^{y_i}}{\sum_{j=1}^n e^{y_j}} = \log \frac{e^{y_i}}{\sum_{j=1}^n e^{y_j}}$$

$$\implies -\sum_{i=1}^m (y_{\text{true}})_i \log(\frac{e^{y_i}}{\sum_{j=1}^m e^{y_j}})$$

$$\implies L(\vec{y} - a\mathbf{1}, \vec{y_{\text{true}}}) = L(\vec{y}, \vec{y_{\text{true}}})$$

(b) Show that for a monotonic and translation-invariant loss L:

$$\max_{\|x - x'\|_\infty \leq \epsilon} = L(f_\theta(\vec{x'}, \vec{y_{\text{true}}}) \leq L(-J_\epsilon(\vec{x}), g_\theta(\vec{y_{\text{true}}\mathbf{1}^\top} - I)), \vec{y_{\text{true}}})$$

3.11 The Fenchel Conjugate of 1_{Z_j}

Solution:

(10):

(a) recall fenchel conjugate of f(x) is defined as

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x \rangle - f(x)\}$$

and the characteristic function for Z is defined as

$$\mathbf{1}_{\mathcal{Z}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{Z} \\ \infty & \text{otherwise} \end{cases}$$

when $l_j \geq \mathcal{Z}_{2j} \leq u_j$ and $l_j \leq u_j \leq 0$

$$\mathbf{1}^*_{\mathcal{Z}}(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x\rangle - \mathbf{1}_{\mathcal{Z}}(x)\} = \sup_{x \in \mathcal{Z}} \{\langle y, x\rangle - 0\}$$

given $y = (\widehat{\nu}, -\nu)$

$$\mathbf{1}^*_{\mathcal{Z}_j}(\widehat{\nu}, -\nu) = \sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{\widehat{\nu} z_{2j} - \nu \widehat{z}_{2j}\}$$

given $\widehat{z}_{2j} \leq 0$ and $z_{2j} = \text{relu}(\widehat{z}_{2j})$, $z_{2j} = 0$, so

$$\mathbf{1}^*_{\mathcal{Z}_j}(\widehat{\nu}, -\nu) = \sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{-\nu \widehat{z}_{2j}\}$$

When $\nu = 0$, $\sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{-\nu \widehat{z}_{2j}\} = 0$

Otherwise, $-\nu \widehat{z}_{2j}$ can be any arbitrary number so $\sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{-\nu \widehat{z}_{2j}\} = \infty$

(b)

Same steps as (a), same steps to derive as part(a), where we get

$$\mathbf{1}^*_{\mathcal{Z}_j}(\widehat{\nu}, -\nu) = \sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{-\nu \widehat{z}_{2j}\}$$

but in this case, $\widehat{z}_{2j} \leq 0$ and $z_{2j} = \text{relu}(\widehat{z}_{2j})$, $z_{2j} = \widehat{z}_{2j}$ so

$$\mathbf{1}^*_{\mathcal{Z}_j}(\widehat{\nu}, -\nu) = \sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{z_{2j}(\widehat{\nu} - \nu)\}$$

its obvious that when $\nu = \widehat{\nu}$, sup is 0 and ∞ otherwise

(11) From derivation in previous part, fenchel conjugate of indicator function of set $\mathcal{Z}_j$ is defined as

$$\mathbf{1}^*_{\mathcal{Z}_j}(\widehat{\nu}, -\nu) = \sup_{(z_{2j}, \widehat{z}_{2j}) \in \mathcal{Z}_j} \{\widehat{\nu} z_{2j} - \nu \widehat{z}_{2j}\}$$

consider the three vertices, $(\widehat{\mathcal{Z}}_{2j}, \mathcal{Z}_{2j})$: $A = (l_j, 0), B = (0, 0), C = (u_j, u_j)$

At vertex A $(z_{2j} = 0, \widehat{z}_{2j} = l_j)$:

$$\widehat{\nu} \cdot 0 - \nu l_j = -\nu l_j$$

At vertex B $(z_{2j} = 0, \hat{z}_{2j} = 0)$:

$$\widehat{\nu} \cdot 0 - \nu \cdot 0 = 0$$

At vertex C $(z_{2j} = u_j, \widehat{z}_{2j} = u_j)$:

$$\widehat{\nu} u_j - \nu u_j = (\widehat{\nu} - \nu) u_j$$

to make sure the supremum does not go unbounded, consider the line that connects vertex A and C

$$-u_j\widehat{y} + (u_j - l_j)\, y = -u_j l_j$$

equivalently, we can set

$$\nu = \frac{\widehat{\nu} u_j}{u_j - l_j}$$

which is the slope of this line, to make sure that the optimal pairs line up along the boundary

4 Extensions

4.1 Main Idea

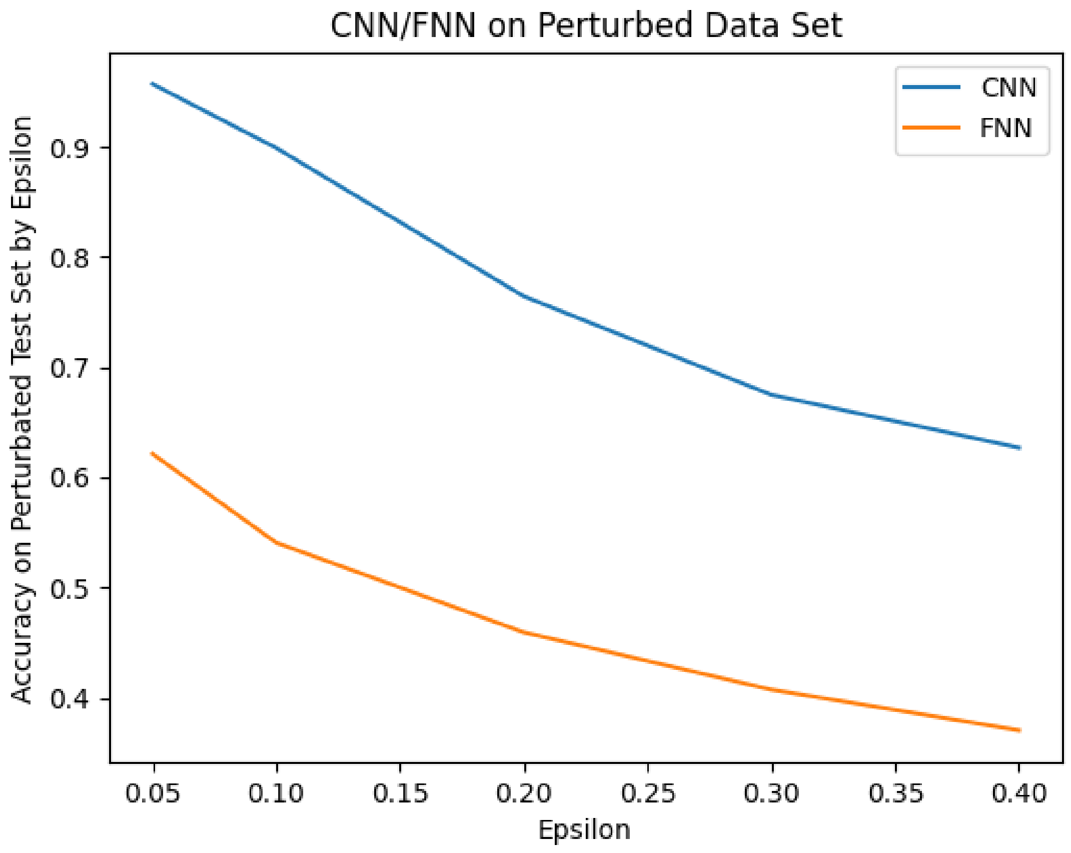
Instead of using a fully connected neural network for image classification, use convolutional neural network to improve the robustness of our model. Convolution networks have been proven to perform better than a standard fully connected neural network on unperturbed images. Our goal in this extension is to test the performance of a convolutional neural network on perturbed images, and compare the results. Furthermore, we want to explore the tradeoffs between training time/compute power with test accuracy. Our question then becomes: How does model architecture affect performance against adversarial examples with different amounts of perturbation?

Since the neurons in each CNN layer correspond to a small region the input, only a few neurons will be affected by the perturbation. Moreover, convolutional layers preserve the spatial context of input by considering the relationship between adjacent pixels. Even if one pixel was perturbed, the surrounding pixels could still provide meaningful context for the neuron. Lastly, as the number of layers grow, the neural network can capture more complex features that are less-sensitive to the perturbations.

4.2 Methodology

We will train and classify the MNIST dataset using a fully-connected and convolutional neural network. More specifically, we will use a ResNet18 convolutional neural network from PyTorch, along with a cross entropy loss function. We will use a AdamW optimizer. We will train on 3 epochs. Moreover, we will test both models with varying epsilons on the test set perturbed by FGSM, and compare their test accuracy. Test accuracy perturbed data points will be used as a measure of robustness. We will also use a 3 layer fully connected neural network, trained on the same AdamW optimizer, and train for 3 epochs.

4.3 Results



Here, we see the test accuracy for both the convolutional neural network and fully connected neural network. For small perturbations, like $\epsilon = 0.05$, the CNN has a better performance than the fully connected layer. However, the rate at which the test accuracy is decreasing with increasing epsilon is the same the for both models. Test accuracy for CNN is higher than the test accuracy from FNN for all ϵ, which shows a higher robustness displayed by the CNN. These results were expected, but the computing time for the convolutional neural network was $5x$ the computing time for the fully connected neural network.