Exercise 04 - Solution

Adversarial Defenses and Certification

Reliable and Interpretable Artificial Intelligence
ETH Zurich

Problem 1 (Coding)**.** In this task, you are going to implement adversarial training with PGD (originally introduced in [1]) and an alternative defense.

1. Complete the provided code skeleton in `train.py` to train the network `model` with PGD defense. That is, for data distribution D (the MNIST dataset in our case) and network parameters θ, optimize the following objective:

$$\min_{\theta} \ \mathbb{E}_{(x,y)\sim D}\left[\max_{x' \in \mathbb{B}_{\epsilon}(x)} L(\theta, x', y)\right]. \tag{1}$$

   Here, $\mathbb{B}_{\epsilon}(x) := \{x' \mid \|x - x'\|_{\infty} \leq \epsilon\}$ denotes the ϵ-sized ℓ_{∞}-ball around x. L is the usual classification loss $L(\theta, x', y) := \mathcal{H}(y, f_{\theta}(x'))$, where $f_{\theta} = (\texttt{model} \circ \text{softmax})$ denotes the output distribution of the neural network, and $\mathcal{H}$ the cross entropy[1] between distributions (being a discrete value, we treat y as a one-hot distribution). In PyTorch, you can use `nn.CrossEntropyLoss`[2] to implement L.

   Use PGD to solve the inner optimization problem with $\epsilon = 0.1$, $k = 7$ steps, and $\epsilon_{\text{step}} = 2.5\frac{\epsilon}{k}$. You can reuse your implementation of untargeted PGD from the previous exercise, or create a more efficient (batched) version better suited for training.

   Compare the accuracy results with and without PGD training.

2. The TRADES [2] algorithm minimizes the following objective (see [2] for details):

[1] `https://en.wikipedia.org/wiki/Cross_entropy`
[2] `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`

$$\min_{\theta} \ \mathbb{E}_{(x,y)\sim D}\big[\underbrace{L(\theta, x, y)}_{\text{for accuracy}} + \underbrace{\lambda \max_{x' \in \mathbb{B}_\epsilon(x)} L(\theta, x', f_\theta(x))}_{\text{regularization for robustness}}\big]. \tag{2}$$

Extend your implementation in `train.py` to the TRADES defense. Again, the inner optimization problem is solved using PGD. Use $\lambda = 1.0$ and the same parameters as in the previous task. Compare your results with the previous task.

Note: Here, $L(\theta, x', f_\theta(x))$ still denotes the cross entropy loss. However, PyTorch's `nn.CrossEntropyLoss` cannot be used as it expects one of the distributions to be a one-hot encoding of the label. Instead, you will need to manually implement L. To this end, some existing functions in PyTorch[3] [4] may be useful.

Solution 1. See `train.py`.

Defense	Clean Accuracy	Adversarial Accuracy
None	0.98	0.26
PGD	0.98	0.90
TRADES	0.98	0.89

Table 1: Test set accuracy.

The accuracy obtained with the default parameters and 50 (instead of 10) epochs is shown in table 1. Without any defense, the adversarial accuracy drops significantly, while it remains high with a defense. Note that the PGD defense outperformed TRADES here (higher adversarial accuracy at the same clean accuracy)—MNIST is a very "easy" task where adversarial training does not impact the clean accuracy too much. Note that the model and hyper-parameters used here are far from optimal.

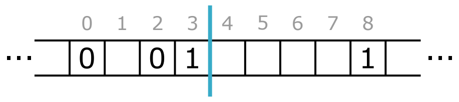
Problem 2 (Alternating optimization)**.** We formalized adversarial defense as the optimization problem in (1). In general, finding the globally optimal parameters θ^* is hard due to the nesting of maximization and minimization. In the lecture, we considered a method that *approximates* θ^* by individually solving the outer and inner optimization problem in alternation. In this task, you will show that this approach can result in a local minimum that is not globally optimal.

Consider the basic case in one dimension where the data is of the form (x, y) with discretized $x \in \mathbb{Z}$ and label $y \in \{0, 1\}$. Assume a very simple classifier ρ in one parameter

[3] `https://pytorch.org/docs/stable/nn.html#loss-functions`

[4] `https://pytorch.org/docs/stable/generated/torch.nn.LogSoftmax.html#torch.nn.LogSoftmax`

$\theta \in \mathbb{Z}$, which classifies a point x as $\rho(\theta, x) = [x \geq \theta]$, where $[\cdot]$ is the Iverson bracket. [5] Consider a scenario where $D = \{(0,0), (2,0), (3,1), (8,1)\}$ as illustrated below:



For a datapoint (x, y), we define the loss $L(\theta, x, y) := [\rho(\theta, x) \neq y]$. Assuming all data points in D are equally likely, the expected loss of a model $\rho(\theta, \cdot)$ is the fraction of misclassified points in D. In our example, the model for $\theta = 4$ (indicated as a blue line) has expected loss $\frac{1}{4}$ as it misclassifies one out of four points. We set $\epsilon = 2$ for the adversarial region $\mathbb{B}_\epsilon(x)$ to denote the set of integers with distance at most 2 to x. For example, point $(8, 1)$ can be perturbed to $(6, 1)$.

Show how in this scenario, alternating optimization of (1) can result in a local optimum that is strictly worse than the global optimum.

Solution 2. In Fig. 1, we show how alternating optimization leads to a suboptimal local optimum.

Round 1, outer problem: First, we find the optimal parameter θ minimizing the expected loss for the initial data points. The optimal parameter is $\theta = 3$ (a).

Round 1, inner problem: We maximize the expected loss for the given parameter $\theta = 3$ by perturbing data points as shown in (b). The expected loss for $\theta = 3$ is thus $\frac{1}{2}$.

Round 2, outer problem: Optimizing θ over the perturbed points, one possible solution is $\theta = 1$ (c).

Round 2, inner problem: The expected loss for the given θ is maximized when perturbing point $(0, 0)$ as shown in (d). The expected loss is for $\theta = 1$ is again $\frac{1}{2}$.

Round 3, outer problem: We optimize θ over the newly perturbed points to again obtain $\theta = 3$, as in round 1. The process continues back at (b) and the expected loss gets stuck at the local optimum $\frac{1}{2}$.

However, the *global* optimum is achieved for $\theta = 6$ where the expected loss is only $\frac{1}{4}$, see (f). Note that in this case, the adversary cannot increase the loss by perturbing any data points.

[5] `https://en.wikipedia.org/wiki/Iverson_bracket`

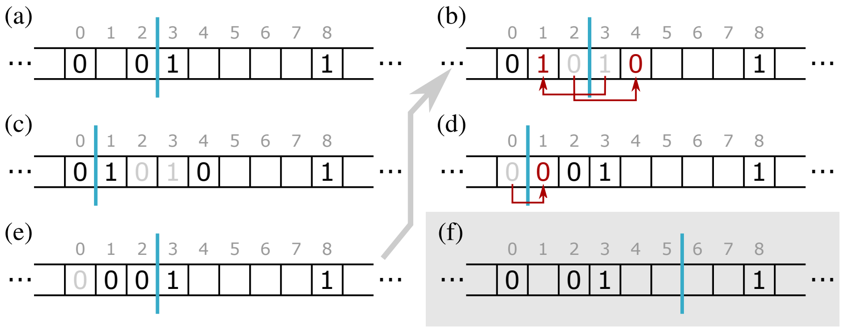


Figure 1: (a–e) Example stuck in a local optimum. Left: Optimizing the parameter θ for given data points. Right: Adversarial examples maximizing the loss for a given parameter θ. (f) Global optimum.

Problem 3 (Box Transformers)**.** In the lecture, you have seen the dox domain for numerical analysis. For vectors $a, b \in \mathbb{R}^m$ with $\forall i.\ a_i \leq b_i$, the box $[a, b]$ is a hypercube in $\mathbb{R}^m$. We can use abstract transformers to obtain an over-approximation of the behavior of a function. Given a function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ and an input box $[a, b] \subset \mathbb{R}^n$, a sound abstract transformer $f^\sharp$ finds $[c, d] \subset \mathbb{R}^m$ such that $\forall x \in [a, b].\ f(x) \in [c, d]$.

For example, let $x \in [1, 3]$ and $y \in [2, 4]$, and assume we want to approximate the result of $2x - y$. Using the basic abstract transformers from the lecture, we can compute

$$2 \cdot^\sharp [1, 3] -^\sharp [2, 4] \;=\; [2, 6] +^\sharp [-4, -2] \;=\; [-2, 4]$$

and conclude that $2x - y \in [-2, 4]$.

1. Show that the box transformers lose precision, by approximating the outcome of $x - x$ for $x \in [0, 1]$ using the transformers $+^\sharp$ and $-^\sharp$ from the lecture.

2. Prove or disprove: The alternative transformer $[a, b] +' [c, d] = [a + c,\ b + |d|]$ for addition is sound (i.e., the output box is an over-approximation of all possible result values).

3. Prove or disprove: The alternative transformer $[a, b] +'' [c, d] = [-\infty,\ a + b + d]$ for addition is sound.

4. Derive a sound abstract transformer $f^\sharp$ for the function $f(x) := x^2$. That is, derive expressions for g, h such that $[g, h] = f^\sharp([a, b])$ for $a, b \in \mathbb{R}$.

5. Derive a sound abstract transformer $\cdot^\sharp$ for multiplication. That is, derive expressions for g, h such that $[g, h] = [a, b] \cdot^\sharp [c, d]$, where $a, b, c, d \in \mathbb{R}$.

Solution 3.

1. It is $[0,1] -^\sharp [0,1] = [0,1] +^\sharp [-1,0] = [-1,1]$. However, the only possible result of $x - x$ for $x \in [0,1]$ is 0, which could be represented exactly using the more precise interval $[0,0]$.

2. This is true. Let $x \in [a,b]$ and $y \in [c,d]$. Hence, $a \leq x \leq b$ and $c \leq y \leq d$. Now consider the sum $z := x+y$. It is $a+c \leq z$ and $z \leq b+d \leq b+|d|$, because $d \leq |d|$. Therefore, $z \in [a+c,\ b+|d|]$.

3. This is not true. Consider $[-1,-1] +'' [0,0] = [-\infty,-2]$, which does not include value -1 produced by $-1+0$.

4. Recall that $a \leq b$. We have to distinguish the three general positions of the interval w.r.t. 0: below ($b \leq 0$), above ($a \geq 0$), or including 0 (otherwise).

$$[g,h] = f^\sharp([a,b]) = \begin{cases} [a^2, b^2] & \text{if } a \geq 0, \\ [b^2, a^2] & \text{else if } b \leq 0, \\ [0, \max(a^2,b^2)] & \text{otherwise.} \end{cases}$$

5. $$[a,b] \cdot^\sharp [c,d] \;=\; [\min(ac,ad,bc,bd),\ \max(ac,ad,bc,bd)]$$

   Note: To see this, perform a case distinction on the signs of a, b, c, and d. The naive solution attempt $[ab, cd]$ is unsound (why?).

References

[1] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". ICLR (2018).

[2] Hongyang Zhang et al. "Theoretically Principled Trade-off between Robustness and Accuracy". ICML (2019).