Foundations of Deep Learning **Fall 2024**

Exercise 11: Adversarial Examples

Lecturer: Aurelien Lucchi

Problem 1 (Projected Gradient Ascent (PGA)):

Projected Gradient Ascent (PGA) is commonly used to generate adversarial examples in machine learning models. In adversarial attacks, the goal is to modify the input $\mathbf{x_0}$ to create a perturbed version $\mathbf{x}_{\text{adv}}$ that maximizes the model's loss, leading to a misclassification. PGA operates by iteratively perturbing the input in the direction that increases the model's loss the most, while keeping the perturbation constrained within a norm ball (typically L_p-norm).

```
Algorithm 1: Projected Gradient Ascent (PGA) for Adversarial Attack [Madry et al.]
  Input: $\mathbf{x}_0$: Original input, $y$: True label, $model$: Classifier model, $\epsilon$: Maximum perturbation, $\alpha$: Step size,
      $num\_iterations$: Number of iterations
  Output: $\mathbf{x}_{\text{adv}}$: Adversarial example
  $\mathbf{x} \leftarrow \mathbf{x}_0$ // Initialize input
  for $i \leftarrow 1$ to $num\_iterations$ do
    $\mathbf{g} \leftarrow \nabla_{\mathbf{x}} \mathcal{L}(model(\mathbf{x}), y)$ // Compute gradient of loss w.r.t. input
    $\mathbf{v} \leftarrow \text{argmax}_{\mathbf{v}:\|\mathbf{v}\|_p \le 1} \mathbf{v}^\top \mathbf{g}$ // Find the maximum ascent direction
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha \cdot \mathbf{v}$ // Update the input using gradient ascent
    $\mathbf{x} \leftarrow \text{project}(\mathbf{x}, \mathbf{x}_0, \epsilon)$ // Project the input back to feasible region
    if $model.predict(\mathbf{x}) \neq y$ then
      break // Stop if the model misclassifies
  return $\mathbf{x}$ // Return the adversarial example
```

```
Algorithm 2: Projection Function
  Input: $\mathbf{x}$: Updated input, $\mathbf{x}_0$: Original input, $\epsilon$: Maximum perturbation
  Output: $\mathbf{x}_{\text{proj}}$: Projected input
  $\mathbf{x}_{\text{proj}} \leftarrow \text{argmin}_{\mathbf{x}^* \in \mathcal{B}^p_\epsilon(\mathbf{x}_0)} \|\mathbf{x}^* - \mathbf{x}\|_p$ // project the input to the $L^p$-ball $\mathcal{B}^p_\epsilon(\mathbf{x}_0) := \{\mathbf{x}^* : \|\mathbf{x}^* - \mathbf{x}_0\|_p \le \epsilon\}$
  return $x_{proj}$
```

a) How do you implement the maximum ascent direction $\mathbf{v} \leftarrow \text{argmax}_{\mathbf{v}:\|\mathbf{v}\|_p \le 1} \mathbf{v}^\top \mathbf{g}$ in Algorithm 1 for $p = 1, 2, \infty$?
Hint: You can apply the following lemma:

Lemma 1 (Optimal Perturbation). *Let $\mathbf{z}$ be an arbitrary non-zero vector, e.g. $\mathbf{z} = \nabla_{\mathbf{x}} \ell_{\text{adv}}$, and let $\mathbf{v}$ be a vector of the same dimension. Then,*

$$\mathbf{v}^* = \underset{\mathbf{v}:||\mathbf{v}||_p \le 1}{\text{argmax}}\, \mathbf{v}^\top \mathbf{g} = \frac{\text{sign}(\mathbf{g}) \odot |\mathbf{g}|^{p^*-1}}{||\mathbf{g}||_{p^*}^{p^*-1}}$$

where $\odot$, $\text{sign}(\cdot)$ and $|\cdot|$ denote elementwise product, sign and absolute value, and p^* is the Hölder conjugate of p, given by $1/p + 1/p^* = 1$. Note that the maximizer $\mathbf{v}^*$ is attained at a $\mathbf{v}$ with $||\mathbf{v}||_p = 1$, since $\mathbf{v}^\top \mathbf{z}$ is linear in $\mathbf{v}$.

b) Write down how you implement the projection function $\mathbf{x}_{\text{proj}} \leftarrow \text{argmin}_{\mathbf{x}^* \in \mathcal{B}^p_\epsilon(\mathbf{x}_0)} \|\mathbf{x}^* - \mathbf{x}\|_p$ in Algorithm 2 for $p = 1, 2, \infty$.

Resources:

Madry A., Makelov A., Schmidt L., Tsipras D., and Vladu A. "*Towards Deep Learning Models Resistant to Adversarial Attacks*" International Conference on Learning Representations. 2018.
https://arxiv.org/abs/1706.06083

Problem 2 (DeepFool Attack (DFA)):

Let $f(\mathbf{x})$ denote a classifier model, where $f(\mathbf{x}) = \text{argmax}_i f_i(\mathbf{x})$ with $f_i(\mathbf{x})$ representing the logits (un-normalized outputs) for class i. The decision boundary between class i and class j is defined by the set of points where the logits for the two classes are equal: $f_i(\mathbf{x}) = f_j(\mathbf{x})$.

The goal of **DeepFool attack** (DFA) is to find the smallest perturbation $\mathbf{v}$ such that the perturbed input $\mathbf{x}_{\text{adv}} = \mathbf{x} + \mathbf{v}$ is classified differently than the original input, i.e., $f(\mathbf{x}_{\text{adv}}) \neq f(\mathbf{x})$. The algorithm approximates the classifier with a linear model at each iteration, assuming the decision boundary is locally linear. It then computes the minimal perturbation needed to cross the boundary and iteratively refines the perturbation until the decision changes. The algorithm stops when the perturbed input crosses the decision boundary, i.e., when the classifier assigns a new label $f(\mathbf{x}_{\text{adv}}) \neq f(\mathbf{x})$.

Algorithm 3: DeepFool Attack [Moosavi-Dezfooli et al.]

```
Input: $\mathbf{x_0}$: Original input, $f$: Classifier model
Output: $\mathbf{x}_{\text{adv}}$: Adversarial example
Initialize:
$\mathbf{x} \leftarrow \mathbf{x_0}$ // Start with original input
while $f(\mathbf{x}) = f(\mathbf{x_0})$ do
    // Find the class closest to the decision boundary
    $j \leftarrow \text{argmin}_{j \neq f(\mathbf{x})} \frac{|f_j(\mathbf{x}) - f_{f(\mathbf{x_0})}(\mathbf{x})|}{\|\nabla f_j(\mathbf{x}) - \nabla f_{f(\mathbf{x_0})}(\mathbf{x})\|_p}$ // Find the nearest decision boundary
    $\mathbf{v} \leftarrow \text{argmin}_{\mathbf{v}} \|\mathbf{v}\|_p$ s.t. $(f_j(\mathbf{x}) - f_{f(\mathbf{x_0})}(\mathbf{x})) + \mathbf{v}^\top \nabla(f_j(\mathbf{x}) - f_{f(\mathbf{x_0})}(\mathbf{x})) = 0$ // Compute perturbation
    $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$ // Update the input with perturbation
return $\mathbf{x}$ // Return adversarial example
```

a) Show that the update rule

$$\mathbf{v} \leftarrow \underset{\mathbf{v}}{\text{argmin}} \|\mathbf{v}\|_p \quad \text{s.t.} \quad (f_j(\mathbf{x}) - f_{f(\mathbf{x_0})}(\mathbf{x})) + \mathbf{v}^\top \nabla(f_j(\mathbf{x}) - f_{f(\mathbf{x_0})}(\mathbf{x})) = 0 \tag{1}$$

can be implemented as:

$$\mathbf{v} \leftarrow \frac{|h(\mathbf{x})|}{\|\nabla h(\mathbf{x})\|_{p^*}} \cdot \frac{|\nabla h(\mathbf{x})|^{p^*-1} \odot \text{sign}(\nabla h(\mathbf{x}))}{\|\nabla h(\mathbf{x})\|_{p^*}^{p^*-1}} \tag{2}$$

where $h := f_j - f_{f(\mathbf{x_0})}$ and p^* is the Hölder conjugate of p, given by $1/p + 1/p^* = 1$.

b) We can define the *linearized binary classifier* $\bar{f}$ between classes i and j at $\mathbf{x}$ as

$$\bar{f}_i(\mathbf{v}) - \bar{f}_j(\mathbf{v}) = f_i(\mathbf{x}) - f_j(\mathbf{x}) + \mathbf{v}^\top(\nabla f_i(\mathbf{x}) - \nabla f_j(\mathbf{x}))$$

which takes positive or negative values depending on which side of the *linearized (or affine) decision boundary* $\{\mathbf{x} : \bar{f}_i(\mathbf{x}) = \bar{f}_j(\mathbf{x})\}$ the perturbation $\mathbf{v}$ lies. Verify that $\frac{|h(\mathbf{x})|}{\|\nabla h(\mathbf{x})\|_{p^*}}$ is the distance to the linearized binary classifier's decision boundary, and that $\frac{|\nabla h(\mathbf{x})|^{p^*-1} \odot \text{sign}(\nabla h(\mathbf{x}))}{\|\nabla h(\mathbf{x})\|_{p^*}^{p^*-1}}$ is the normalized (L_p-norm equal to one) direction of steepest ascent.

c) How do you further implement the update rule (2) for $p = 1, 2, \infty$?

d) Note that PGA in Algorithm 1 has maximum perturbation constraint ϵ while DFA in Algorithm 3 does not, meaning the former is a constrained adversarial perturbation while the latter an unconstrained one. Discuss advantages and disadvantages of constrained vs. unconstrained adversarial perturbations.

e) The "ideal" constraint that we would like to impose on adversarial perturbations is that they are imperceptible to a human. The norm constraints that are used in the above iterative adversarial attacks are just proxies for this imperceptibility constraint. Discuss the limitations of norm-constrained perturbations.

Resources:

Moosavi-Dezfooli S., Fawzi A., and Frossard A.. "*Deepfool: a simple and accurate method to fool deep neural networks.*" Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

`https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Moosavi-Dezfooli_DeepFool_A_Simple_CVPR_2016_paper.pdf`