Exercise 03

Adversarial Examples

Reliable and Interpretable Artificial Intelligence
ETH Zurich

Problem 1 (Coding)**.** This task can be done either in `task.py` or `task.ipynb` (whichever you prefer). You are provided an already trained MNIST classifier `model`. Instead of this provided model you can also use your own MNIST classifier.

1. Implement targeted FGSM. The signature of your implementation should be similar to `fgsm_targeted(model, x, target, eps)`. Your implementation should also clamp back to the image domain (i.e. $[0, 1]^{28\times 28}$ for MNIST). Feel free to extend the signature as needed.

2. Implement untargeted FGSM. Similar to the targeted attack, but instead of a `target` the correct `label` is passed.

3. Implement iterated, projected FGSM (also known as PGD attack; as described by [1]) in both targeted and untargeted setting. This algorithm performs k iterations of FGSM with perturbation magnitude ϵ_s each. After each step the current solution is projected back to the ϵ sized ℓ_∞-ball around the initial starting input. Note that projection to the ℓ_∞-ball can be obtained by just clipping values. Use your solutions from the previous two tasks. The signature should look like `pgd_targeted(model, x, target, k, eps, eps_step)` where `eps` is the size of the ℓ_∞-ball to be projected on and `eps_step` is the size for an individual FGSM step. Again you should clip to the image domain. The signature for the untargeted case has the same arguments.

4. **Optional:** FGSM and PGD attacked images for MNIST do not look very impressive as the perturbation is clearly visible. Run your attacks also for CIFAR-10. If you implemented your attacks correctly you should not need to change anything but datapoint `x` and the `model` passed to the function. You will see that there the attacks can be hidden much more in the image.

Problem 2 (Projection onto ℓ_2-ball)**.** The Euclidean projection z of a point y onto the ϵ-size ℓ_p-ball around x is defined as (note the ℓ_2 norm):

$$z = \underset{x' \text{ s.t. } \|x'-x\|_p \leq \epsilon}{\arg\min} \|x' - y\|_2$$

In general, this is a hard problem and closed form solutions are only known for few p. For example, we have considered projections for $p = \infty$ in the lecture. Here, we are investigating the case for $p = 2$.

1. Derive the closed form solution of projecting a point y onto the ϵ-ℓ_2-ball around a point x.

2. Prove that in 2 dimensions, your closed form solution z is correct, i.e., show that there exists no point $q \neq z$ in the ϵ-ℓ_2-ball around x that is closer to y than z. *Hint:* Assume for the sake of contradiction that there exists such a point q. Use the triangle inequality.

Problem 3 (Extending PGD)**.** In this problem you are asked to extend the PGD algorithm to heuristically perform targeted attacks with respect to the ℓ_p-norm (assume that $2 \leq p < \infty$). That is, it should produce an output $\boldsymbol{x}'$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \epsilon$. Complete the implementation of `MyPGD` below by performing the following steps:

```
def MyPGD(x, t, k, ϵ, ϵ_s)
    ................................................;
    for i ← {1, ..., k} do
        g ← ∇_x' loss_t(f(x'));
        ............................................;
        ............................................;
        ............................................;
        ............................................;
    end
    return x';
end
```

1. Generate a random point $\boldsymbol{x}'$ in the ℓ_p-ball of size ϵ around $\boldsymbol{x}$.

2. Perform update steps that are similar to FGSM and move by ϵ_s (with respect to the ℓ_p-norm) according to the direction of the (normalized) gradient $\boldsymbol{g}$.

3. Perform projection steps onto the ϵ sized ℓ_p-ball.

Note: Perform the projection approximately, following the same strategy as in Problem 2. There is no general closed form rule to project on ℓ_p-boxes. Aside from special cases ($p = 1, 2, \infty$), these projections cannot be computed directly and the projection step typically needs to be treated as an optimization problem itself. For $p = 2$ the resulting PGD attack which you will derive performs the correct projection (as you have shown in Problem 2) and is used like this in practice.

References

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