LINMA2300: Project 2
Derivative free optimization for adversarial attacks on neural classifier

Prof. Geovani Grapiglia, Bastien Massion, Nicolas Mil-Homens Cavaco

29 November 2024, v2

1 Context

Consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with N images $x_i \in \mathcal{X} \subseteq \mathbb{R}^{c\times w\times h}$ and labels $y_i \in \mathcal{K} = \{0, \cdots, k-1\}$, where c is the number of channels, w, h are the number of pixels along the width and height of the images, and k is the number of classes. We build a classifier $\mathcal{C}_\theta : \mathcal{X} \to \mathcal{K}$ with parameters θ that, for an image x, outputs a predicted label $\hat{y} \in \mathcal{K}$. The classifier can be decomposed into three parts. First, a model $f_\theta : \mathcal{X} \to \mathbb{R}^k$ which associates a score $s_j \in \mathbb{R}$ for the likelihood of the image to belong to each class $j \in \mathcal{K}$:

$$f_\theta(x) = s \in \mathbb{R}^k.$$

The second part of the classifier $g : \mathbb{R}^k \to [0,1]^k$ normalizes the scores s_j to predicted probabilities of x to belong to class $j \in \mathcal{K}$. The normalization usually consists in a *softmax* operation:

$$[g(s)]_j = \frac{e^{s_j}}{\sum_{\ell=0}^{k-1} e^{s_\ell}}.$$

Finally, the third part selects the label with the highest probability

$$\hat{y} = \underset{j\in\mathcal{K}}{\operatorname{argmax}}[g(f_\theta(x))]_j = \underset{j\in\mathcal{K}}{\operatorname{argmax}} \frac{e^{[f_\theta(x)]_j}}{\sum_{\ell=0}^{k-1} e^{[f_\theta(x)]_\ell}} = \mathcal{C}_\theta(x).$$

In the modern-day era of *machine learning*, the classifier (in particular the model f_θ) is typically a neural network, such as a convolutional neural network. The model parameters θ are often obtained by the minimization of a cross-entropy loss function by an adaptive stochastic gradient descent algorithm (and backpropagation) such as Adam. The goal of such a classifier is to accurately predict the label such that $y = \hat{y}$ for as many images as possible.

However, even a well-trained classifier can have weaknesses. One such weakness is the vulnerability against *adversarial attacks*. Adversarial attacks are perturbations $d \in \mathcal{X}$ such that $\mathcal{C}_\theta(x) \neq \mathcal{C}_\theta(x+d)$, often applied on images that were successfully classified. Typically, these perturbations may be very small in amplitude, meaning that for some choice of norm $\|\cdot\|$ (for example $\|\cdot\|_\infty$), there exists some small $\epsilon > 0$ such that $\|d\| \leqslant \epsilon$, providing perturbations that can seem completely imperceptible to human eye.

A possible way to construct adversarial attacks is by using derivative-free optimization methods. Indeed, such methods consider the classifier as a black-box model, simply outputting the probability distribution $g(f_\theta(x)) \in [0,1]^k$, for some image $x \in \mathcal{X}$. In this project, we consider *targeted* adversarial attacks, which means that the goal of the attacker is to perturb the image such that the classifier predicts a wrong predetermined class $t \in \mathcal{K}$. Thus, it aims to maximize the probability associated with the target t while minimizing the probabilities of the other classes. This can be done through the following optimization problem [1]:

$$\begin{aligned} \min_{d\in\mathbb{R}^{c\times w\times h}} \quad & F(d) := \log\left(\textstyle\sum_{j\neq t}[g(f_\theta(x+d))]_j\right) - \log\left([g(f_\theta(x+d))]_t\right) \\ & \text{s.t.} \quad \begin{cases} x+d \in \mathcal{X} \\ \|d\|_\infty \leqslant \epsilon. \end{cases} \end{aligned} \tag{1}$$

2 Dataset

Like in the previous project, you will use the dataset CIFAR-10: RGB images ($c = 3$) of $w \times h = p \times p$ pixels, with $p = 32$. The dataset will be downloaded through the `torchvision` package, an extension to `torch`, one of the most widely used library for working with neural networks in Python [2]. The dataset will be downloaded and managed automatically by the code provided in the Notebook for this project. In the Notebook, the dataset is split into two parts: the training set contains $N_{\text{train}} = 50000$ images and the test set $N_{\text{test}} = 10000$. Each image x is represented as an element of $\mathcal{X} = [-1,1]^{3\times32\times32}$. Moreover, each image is associated with a label $y \in \mathcal{K} = \{0, \cdots, 9\}$ ($k = 10$), which respectively correspond to {`airplane`, `automobile`, `bird`, `cat`, `deer`, `dog`, `frog`, `horse`, `ship`, `truck`}. For the training of your neural network classifier, the data are loaded in batches of size $b = 64$.

3 Part 1: CNN Classifier

The model $f_\theta(\cdot)$ is a Convolutional Neural Network (CNN). The main feature of this architecture is the succession of convolutional layers, as illustrated in Figure 1. In neural networks, each layer transforms its input into a collection of representations (called *channels*) representing the image's high- or low-level features.

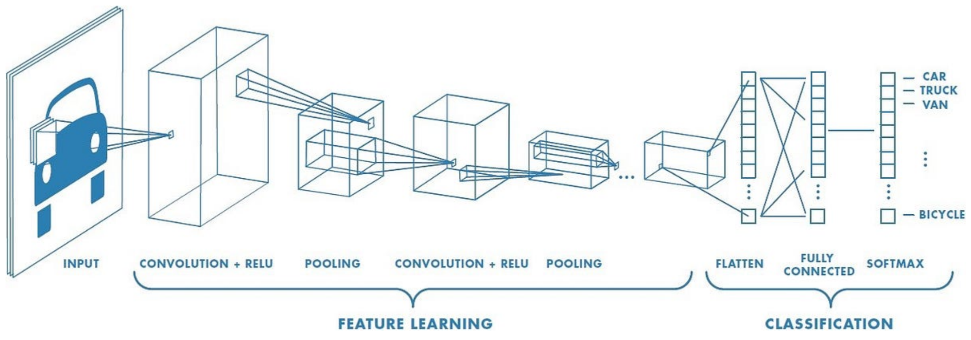

Figure 1: Convolutional Neural Network conceptually

A small but somewhat effective CNN architecture is proposed in the Notebook. You might try to modify it to improve results, but it is not necessary for the completion of this project: you could keep it as it is. If you modify it, make sure to detail your changes in the report. The implementation is done in Python with `torch` (aka PyTorch). One main goal of this part of the project is to introduce you to this powerful and widely-used library.

The hidden layers transformations are parameterised. Good values for these parameters (*weights*) can thus be learned through the training, i.e., the optimization procedure. For the classification task on a given dataset, the *cross-entropy loss* is used: it aims to minimize the surprise of an image being in a given class, computed from the probabilities of the image being in this class:

$$\mathcal{L}(\theta; \mathcal{D}) = -\frac{1}{N}\sum_{i=1}^{N} \log\left([g(f_\theta(x_i))]_{y_i}\right).$$

To minimize this loss function, you will use first-order (gradient) methods. In particular, the methods are batch-stochastic and often adaptive: SGD with momentum, Adam, AdaGrad and RMSProp. The dataset is divided into batches of size b because it would be too costly to compute the full gradient on the whole dataset. The gradient is automatically computed across the whole network thanks to the backpropagation algorithm. The adaptive part of the algorithm comes from the automatic adaptation of the stepsize from the previous iterations.

3.1 Training process

1. Code the training process for your classifier using the `torch` package.

3.2 Machine learning optimization algorithms

You will study 4 optimization algorithms dedicated to machine learning: SGD with momentum, Adam, AdaGrad and RMSProp. For each of them:

1. Train the CNN on at least 30 epochs and report the learning curves (evolution of the loss and evolution of the accuracy for both the training set and the test set). Comment on your result.

2. Give intuitive interpretations of the most important hyperparameters. Explain how they should impact the training and how you would proceed to tune them.

3.3 Hyperparameters tuning

3. You now focus solely on Adam. Tune the learning rate of Adam to improve the accuracy on the test set compared to those defined by default in `torch`. Briefly discuss your findings.

3.4 Recommendation

4. Based on your numerical results and possibly theoretical arguments, which optimization method (and hyperparameter values) would you recommend for training the classifier? Briefly, justify your choice.

4 Part 2: Adversarial attack

In this second part of the project, you will rely on derivative-free optimization methods to perform a targeted adversarial attack on the CNN trained with the algorithm you have recommended in Part 1. Consider an image $x \in \mathcal{X}$ of true label $y \in \mathcal{K}$ such that your CNN classifier provides the right label: $\mathcal{C}_\theta(x) = y$, and select a target $t \neq y$.

4.1 Mathematical formulation

Looking at Problem (1), a hard point is to handle the constraints since the DFO algorithms seen in the course are only designed for unconstrained problems. To avoid this difficulty, we will completely neglect the constraints so that (1) becomes an unconstrained optimization problem.

1. Consider the objective function $F(d)$ of Problem (1). Show that $F(d)$ is unbounded from below.

   Hint: $\sum_{j=0}^{k-1} [g(f_\theta(x+d))]_j = 1$ *for all* $x, d \in \mathbb{R}^{c\times w\times h}$.

To recover an optimization problem with a well-defined solution, we propose to remove the first term of $F(d)$ so that the unconstrained optimization problem simply becomes:

$$\min_{d \in \mathbb{R}^{c \times w \times h}} -\log([g(f_\theta(x+d))]_t).$$

The minimization of the removed term thus becomes implicit due to $\sum_{j=0}^{k-1}[g(f_\theta(x+d))]_j = 1$. To apply DFO methods, the objective function has to depend on a variable in vector form, so we define a new variable by vectorization: $\tilde{d} = \mathrm{vec}(d) \in \mathbb{R}^n$, where $n = cwh$. One can of course find the image back: $d = \mathrm{im}(\tilde{d}) \in \mathbb{R}^{c \times w \times h}$. This leads to the final problem

$$\min_{\tilde{d} \in \mathbb{R}^n} \widetilde{F}(\tilde{d}) := -\log([g(f_\theta(x + \mathrm{im}(\tilde{d})))]_t). \tag{2}$$

2. Show that this new loss function is bounded from below. Give the minimum value of the function and the condition that any minimizer should satisfy.

3. Suppose now that you aim to solve the constrained Problem (1) with derivative-free optimization. What strategy would you use to manage the constraints?

   Remark: We expect you to provide at least one reference from the literature. You are not asked to solve Problem (1) *in your code.*

4.2 Derivative Free Optimization

Focus on Problem 2.

4. Implement the 4 derivative-free optimization algorithms seen in the Lectures: Random Finite Differences (RFD) from Lecture 6, Stochastic Three Points (STP) from Lecture 7, Derivative-Free Trust-Region (DFTR) and the Derivative-Free Trust-Region with Random Subspace Technique (DFTR-RST) from Lecture 8 by using finite differences to approximate the gradient, BFGS to approximate the Hessian, Cauchy's step to find the direction, and a gaussian random matrix Q for the random subspace technique (cf. TP7).

5. Apply each algorithm (use $\tilde{d}_0 = 0_n$ for initialization) to solve (2) on one image x of your choice from the test set that is correctly labelled by your CNN classifier initially: $\mathcal{C}_\theta(x) = y$. Based on your theoretical knowledge of these algorithms, explain what the hyperparameters represent and how you would proceed to tune them.

   Hint: Tuning the hyperparameters for real is unnecessary for completing the question.

   Hint 2: You do not have access to the Lipschitz constant of $\nabla \widetilde{F}(\tilde{d})$, so you do not need to find it explicitly and it is considered as a hyperparameter of the algorithms.

6. Compare the results obtained by each method. Do they succeed in bringing the CNN classifier to the target: $\mathcal{C}_\theta(x + d) = t$?

7. Show the evolution of the modified objective function $\widetilde{F}(\tilde{d})$ against the number of iterations for each method on the same plot. Is it in accordance with what you expect theoretically?

8. Is it fair to compare algorithms with respect to their number of iterations? If no, propose a new metric to compare algorithms more fairly, and explain why it is more fair.

4.3 Recommendation

9. Based on your results, which method would you recommend? Briefly, justify your choice.

10. For the method you recommend, study the evolution of the visual quality of the image $x + d$, the evolution of the amplitude of the noise d and the evolution of the predicted class $\mathcal{C}_\theta(x+d)$ with respect to the number of function evaluations. Answer this question for 2 additional images x_1, x_2 of your choice (correctly classified by your CNN) from the test set and possibly associated with two other targets t_1, t_2 of your choice. Comment on your results.

5 Practical information

- *Group:* Make groups of 2 on the Moodle activity "Group choice for Project 2". Register as soon as possible.
- *Deadline:* **Friday 20th December 2024, 23:59**, in the "Project 2" activity on Moodle.
- *Material:* This assignment and one Jupyter Notebook to complete.
- *Deliverables:* Per group, one report in PDF format **.pdf** and one completed Jupyter Notebook in **.ipynb** format. If your Notebook has dependencies with external files (such as images or Python files **.py**), then the latter should be delivered too.
- *Report:*
  - 5 pages maximum, images included (4 pages should be enough), bibliography not included.
  - The font size should be 11 (single column) or 10 (double columns).
  - Answer all questions and subquestions.
  - Use Figures and Tables to communicate important results. Each Figure or Table should only answer one question or emphasize one aspect of your results.
  - The report should be understandable for a person with the background of someone starting a master's degree as a civil engineer in applied mathematics.

- *References:* You might be interested in looking up external resources (scientific articles, blogs, online implementations, ...). Cite them correctly and include a bibliography at the end of your report.
- *AI:* Using AI writing/coding assistant (such as Chat GPT) is allowed, but you have to add a section at the end of the report detailing in which clever way you used this tool.
- *Q&A:* Two Q&A sessions will take place on:
  - **Wednesday 11th of December 2024, 08:30-10:30**, in BARB13 (during the exercise session).
  - **Friday 13th of December 2024, 10:45-12:45**, in BARB03 (during the exercise session).
- *Contact:* As a last resort, you can send your questions and remarks directly to Bastien Massion (`bastien.massion@uclouvain.be`) and Nicolas Mil-Homens Cavaco (`nicolas.mil-homens@uclouvain.be`).

References

[1] Giuseppe Ughi, Vinayak Abrol, & Jared Tanner. (2020). An Empirical Study of Derivative-Free-Optimization Algorithms for Targeted Black-Box Attacks in Deep Neural Networks.

[2] `https://pytorch.org/`, The Linux Foundation, last access: 30/11/24.