Zero-th order Frank Wolfe methods for adversarial attacks

Jacopo Magliani, Stefano Minto

September 16, 2023

Abstract

An adversarial attack is a tool used to fool a trained Machine Learning Model leading it to mispredict by injecting noise in the input data. In this paper we focus on implement the algorithms Frank Wolfe Black Box Attack (FWBB) and Accelerated Stochastic Zeroth-Order Frank-Wolfe (Acc-SZOFW) to fool a trained deep model on two datasets of images while not creating an exessive distorsion from the original input data.

1 Introduction

Deep Neural Networks (DNNs) are widely used in many different areas of application such as image classification, speech recognition and object detection. However recent studies have shown that these models, although they have been trained properly and can obtain high precision on new data, can be easily fooled by injecting even slight noise into the input data, leading to a slightly or completely incorrect output. This injection of perturbation into the input data is called adversary example, and can be extremely difficult to recognize it has happened since the modification can appear inconsistent to the human eye but can easily mislead a well-trained image classifier towards misclassification. Depending on how much information of the targeted model they have access to, adversarial attacks can be divided into two categories:

1. **white-box attack** : the adversary has full access the structure of the target model, allowing to perform back-propagation in order to calculate the gradient of the loss function and modify the input data. However, this method tends to generate adversarial examples near or upon the boundary of the perturbation set, leading to a large distorsion in the resulting adversarial examples.

2. **black-box attack** : the adversary can only access the input and output of the model but not in its internal configuration. Since we haven't full access to the model we need to perform a gradient estimation and this may lead to a big number of required queries to perform a successful black-box attack, unfeasible when data dimension d is high since it would take $O(d)$ queries to perform one full estimation and results in inefficient attacks.

2 Related Work

The first algorithm we have implemented is FWBB[1], a variant of the Frank Wolfe algorithm projection-free with momentum mechanism, while the second algorithm Acc-SZOFW[2] is based on the variance reduced technique of SPIDER/SpiderBoost and a novel momentum accelerated technique.

3 Problem formulation

Adversarial Attacks can be divided into two categories: untargeted attacks aim to turn the prediction into any incorrect label, while targeted attack push the classifier to mislead to a specific target class. In our project we focus on targeted attacks.
We define the loss function of the targeted Deep Neural Network (DNN) as $f(x) = l(x, y_{tar})$ with an input $x \epsilon R^d$ and a corresponding target class y_{tar}. Then the targeted attack problem can be formulated as:

$$min_x f(x)$$

$$\text{subject to } ||x - x_{ori}||_p \leq \epsilon$$

where the constraint set $X = \{x| \parallel x - x_{ori} \parallel_p \leq \epsilon\}$ is a bounded convex set with $p \geq 1$ and allows us to produce a modified input which does not differ too much from the original one.

A more complete definition of the problem is reported in the paper [2] that introduces Acc-SZOFW:

$$min_{x \epsilon X} f(x) = \begin{cases} E_\xi[f(x;\xi)] & \text{(stochastic)} \\ \frac{1}{n}\sum_{i=1}^{n} f_i(x) & \text{(finite-sum)} \end{cases} \tag{1}$$

where ξ is a random variable following an unknown distribution. When $f(x)$ denotes the expected risk function the problem 1 is a stochastic problem, when $f(x)$ denotes the empirical risk function it will be seen as a finite-sum problem.

4 Frank Wolfe method

The problem 1 appears in many machine learning models and for solving it one common approach is the projected gradient method, which main phases are optimizing the loss function in the unconstrained space and then projecting the solution in the constrained space X. This projection can be quite expensive to compute in many constrained sets, and the Frank-Wolfe method can be a valid alternative. Introduced in the 1950s, it is a projection-free method that only needs to compute a linear operator instead of projection at each iteration. Thanks to its faster projection-free optmization and the ability to keep the iterates in the constrained set, the Frank Wolfe algorithm and its variants are popular in many applications and can even deal with nonconvex problems as adversarial attacks.

The algorithm 1 starts with an initial feasible solution, compute the gradient of the objective function at the current solution, finds the direction in which the objective function decreases the most and uses it to update the solution by moving in the chosen direction with the determined step size. It repeats these steps until a convergence criterion is met, such as reaching a specified tolerance level or a maximum number of iterations.

Algorithm 1 Frank-Wolfe method

Choose a point $x_1 \in C$ constrained set
for k = 1, ... **do**
  Set $\widehat{x}_k = Argmin_{x \in C} \nabla f(x_k)^T (x - x_k)$
  If $\widehat{x}_k$satisfies some specific condition then STOP
  Set $x_{k+1} = x_k + \alpha_k(\widehat{x}_k - x_k)$ with $\alpha_k \in [0, 1]$ suitably chosen step size
end for

In the paper [1] there are both the white-box and the black-box versions of Frank Wolfe that we report in 2 and 3

Algorithm 2 Frank-Wolfe White Box Attack Algorithm

Input: number of iterations T, step sizes γ_t;
$x_0 = x_{ori}, m_{-1} = \nabla f(x_0)$
for t = 0, ...T - 1 **do**
  $m_t = \beta * m_{t-1} + (1 - \beta) * \nabla f(x_t)$
  $v_t = argmin_{x \in X} \langle x, m_t \rangle //LMO$
  $d_t = v_t - x_t$
  $x_{t+1} = x_t + \gamma_t d_t$
end for
output: x_T

The main difference of algorithm 2 with the original 1 is the addition of a momentum term that stabilizes the direction and leads to an accelerated convergence.

The Linear Minimization Oracle finds the x that minimizes the inner product between x and the gradient of the objective function, obtaining the direction used to make progress towards the optimal solution. Since we are dealing with a constrained set we can write the update of the iterate as:

$$x_{t+1} = x_t - \gamma_t \epsilon * sign(m_t) - \gamma_t(x_t - x_{ori}) \tag{2}$$

The term $-\gamma_t(x_t - x_{ori})$ enforces x_t to be close to x_{ori} for all $t = 1, ...T$, forcing the adversarial examples to have a small distorsion.

Algorithm 3 Frank-Wolfe Black Box Attack Algorithm

Input: number of iterations T, step sizes γ_t, sample size for gradient estimation b, sampling parameter δ;
$x_0 = x_{ori}, m_{-1}$ =GRAD_EST(x_0, b, δ)
for t = 0, ...T - 1 **do**
  q_t = GRAD_EST (x_t, b, δ)
  $m_t = \beta * m_{t-1} + (1 - \beta) * q_t$
  $v_t = argmin_{v \in X} \langle v, m_t \rangle$
  $d_t = v_t - x_t$
  $x_{t+1} = x_t + \gamma_t d_t$
end for
output: x_T

We use equation 2 even in the Black-box setting since we are still dealing with the constrained set. In addition, in the algorithm 3 there is an extra gradient estimation step since the algorithm does not have access to the structure of the model. As in many other zeroth-order optimization algorithms, algorithm 4 uses symmetric finite differences to estimate the gradient, getting rid of the dependence on back-propagation in white-box setting. There are two options: using vectors uniformly sampled from Euclidean unit sphere or uniformly sampled from standard multivariate Gaussian distribution.

Algorithm 4 GRAD_EST(x,b,δ)

q = 0
for i = 1, ...b **do**
  option I: sample u_i uniformly from the Euclidean Unit Sphere with $||u_i||_2 = 1$
  $q = q + \frac{d}{2\delta b}(f(x + \delta u_i) - f(x - \delta u_i))u_i$
  option II: sample u_i uniformly from the standard Gaussian Distribution $N(0, 1)$
  $q = q + \frac{1}{2\delta b}(f(x + \delta u_i) - f(x - \delta u_i))u_i$
end for
return: q

5 Accelerated stochastic zero order Frank Wolfe method

FWBB needs to compute the estimated gradient at each iteration and can suffer from high function query complexity in solving the problem 1, especially in high-dimension settings. Because of this the authors of [2] propose a class of accelerated zeroth-order Frank-Wolfe methods: they developed the Accelerated Stochastic zeroth-order Frank Wolfe 5 (Acc-SWFW) based on the variance reduced technique of SPIDER/SpiderBoost and a novel momentum accelerated technique, and a novel version that relaxes the large mini-batch size required in Acc-SZOFW*.

In algorithm 5 for the estimate of the gradient we have two options:

1. **UniGE** (uniform smoothing gradient estimator)

$$\widehat{\nabla}_{uni} f_i(x) = \frac{d(f_i(x + \beta u)) - f_i(x)}{\beta} u \tag{3}$$

where $u \in R^d$ is a vector generated from the uniform distribution over the unit sphere and β is a smoothing parameter.

Algorithm 5 Acc-SZOFW Algorithm

Input: Total iterations T, step-sizes $\{\eta_t, \gamma_t \in (0,1)\}_{t=0}^{T-1}$, weighted parameters $\{\alpha_t \in [0,1]\}_{t=0}^{T-1}$, epoch-size q, mini-batch size b, or b_1, b_2
Initialize: $x_0 = y_0 = z_0 \in X$
for t = 0, ...T - 1 **do**
  if mod(t,q) = 0 then **then**
    For the finite-sum setting, compute $v_t = \widehat{\nabla}_{coo} f(z_t) = \frac{1}{n}\sum_{i=1}^{n} \widehat{\nabla}_{coo} f_i(z_t)$
    For the stochastic setting, randomly select b_1 samples $B_1 = \{\xi_1, ...\xi_{b_1}\}$ and compute $v_t = \widehat{\nabla}_{coo} f_{B_1}(z_t)$ or draw i.i.d. $\{u_1, ...u_{b_1}\}$ from uniform distribution over unit sphere, then compute $v_t = \widehat{\nabla}_{uni} f_{B_1}(z_t)$
  else
    For the finite-sum setting randomly select $b = |B|$ samples $B \subseteq \{1, ..n\}$ and compute $v_t = \frac{1}{b}\sum_{j\in B}[\widehat{\nabla}_{coo} f_j(z_t) - \widehat{\nabla}_{coo} f_j(z_{t-1})] + v_{t-1}$
    For the stochastic setting randomly select b_2 samples $B_2 = \{\xi_1, ...\xi_{b_2}\}$ and compute $v_t = \frac{1}{b_2}\sum_{j\in B_2}[\widehat{\nabla}_{coo} f_j(z_t) - \widehat{\nabla}_{coo} f_j(z_{t-1})] + v_{t-1}$ or draw i.i.d. $\{u_i, ...u_{b_2}\}$ from uniform distribution over unit sphere, then $v_t = \frac{1}{b_2}\sum_{j\in B_2}[\widehat{\nabla}_{uni} f_j(z_t) - \widehat{\nabla}_{uni} f_j(z_{t-1})] + v_{t-1}$
  end if
  Optimize $w_t = argmax_{w\in X}\langle w, -v_t\rangle$
  Update $x_{t+1} = x_t + \gamma_t(w_t - x_t)$
  Update $y_{t+1} = z_t + \eta t(w_t - z_t)$
  Update $z_{t+1} = (1-\alpha_{t+1})y_{t+1} + \alpha_{t+1}x_{t+1}$
end for
Output: z_ζ chosen uniformly random from $z_{t=1}^{T}$

2. **CooGE** (coordinate smoothing gradient estimator)

$$\widehat{\nabla}_{coo} f_i(x) = \sum_{j=1}^{d} \frac{f_i(x+\mu_j e_j) - f_i(x-u_j e_j)}{2\mu_j} e_j \tag{4}$$

where μ_j is a coordinate-wise smoothing parameter, and e_j is a basis vector with 1 as its j-th coordinate, and 0 otherwise. Without loss of generality $\mu = \mu_1 = ... = \mu_d$

The authors first propose an accelerated deterministic zeroth-order Frank-Wolfe (Acc-ZO-FW) algorithm to solve the finite-sum problem 1 as a baseline by using the zeroth-order gradient $v_t = \frac{1}{n}\sum_{i=1}^{n} \widehat{\nabla}_{coo} f_i(z_t)$. When the sample size n is large the computation can be very slow and for the stochastic problem 1 we may never obtain the estimate of the gradient. Because of this the authors adopt a stochastic optimization method involving a mini-batch $B \subseteq \{1, 2, ...n\}(b = |B|)$ or $B = \{\xi_1, ..., \xi_b\}$ from the distribution of random variable ξ and can obtain the stochastic-zeroth-order gradient:

$$\widehat{\nabla} f_\beta(x) = \frac{1}{b}\sum_{j\epsilon\beta} \widehat{\nabla} f_j(x) \tag{5}$$

where $\widehat{\nabla} f_j()$ includes$\widehat{\nabla} coof_j()$ and $\widehat{\nabla}_{uni} f_j()$.

But still the stochastic gradient can suffer from large variance and this may result in high function query complexity, so the authors use the variance reduced technique of SPIDER/SpiderBoost to reduce it.

$$v_t = \begin{cases} \frac{1}{b_1}\sum_{i\in\beta_1} \widehat{\nabla} f_i(x_t), \text{ if mod(t,q)} = 0 \\ \frac{1}{b_2}\sum_{i\in\beta_2}(\widehat{\nabla} f_i(x_t) - \nabla f_i(x_{t-1})), \text{ otherwise} \end{cases} \tag{6}$$

Furthermore, in the last lines of 5 the authors adopt a novel momentum accelerated technique introducing two intermediate variables and the algorithm keeps all $\{x, y, z\}$ in the constrained set X. When $\alpha_{t+1} \in (0,1)$ the updating parameter z_t is a linear combination of the previous terms $w_i - z_i (i \leq t)$.

6 Experiments

Our task was developing the codes for the algorithms FW-Black 3 and Acc-ZO-FW 5 and test them on one of the problems reported in Section 6 of [2]:

1. UAP (Universal adversarial perturbation) : for each class develop a universal perturbation and apply it to all the input data that belong to the same class.

2. SAP (Single adversarial perturbation) : for each input data perform a perturbation in order to minimize the loss function.

We chose the second problem. Furthermore, we tried to adapt the code of the stochastic gradient to SAP situation: if in the UAP problem we sample b_1, b_2 images, we instead sample b_1, b_2 pixels of the same image. We imagined that this would have lead to results somewhere in between FWBB and Acc-ZOFW.

For testing the developed algorithms we used the image datasets MNIST (used by both [1] and [2]) and CIFAR10 (used only by [2]). MNIST dataset consists in 28x28 grey-scale images representing a number between 0 and 9, CIFAR10 dataset has 32x32 RGB images representing one of 10 different classes (e.g. airplanes, cars, birds, cats).

The targeted model to fool was initially created following the settings of [1]: a 6 layers CNN with 4 convolutional layers followed by 2 dense layers with max-pooling and ReLU activations applied after each convolutional layer. The model was trained for 10 epochs on the two datasets, obtaining an accuracy of 99.12% on MNIST test set and an accuracy of 70.82% in the CIFAR10 test set. Then we chose 1000 random test images by Mnist and 100 by Cifar10 that were correctly classified by the model and assigned a random wrong target label to each of them. For all the parameters that had to be set, we decided to use the ones reported in the papers obtained after some experiments on different scales.

Alg	d	ϵ	T	γ_t	δ	β	μ	η	b	b_1	b_2	q
FWBB (Mnist)	28	0.3	10	0.8	0.01	0.99			25			
Acc-SZOFW (Mnist)	28	0.3	10	*		*	0.01	*	1	300	20	20
FWBB (CIFAR10)	32	0.1	10	0.8	0.01	0.99			25			
Acc-SZOFW (CIFAR10)	32	0.1	10	*		*	0.01	*	1	300	20	20

In addition we set for Acc-SZOFW $\alpha_t = \frac{1}{t+1}$,$\theta = \frac{1}{(t+1)*(t+2)}$,$\eta = \frac{1}{\sqrt{T}}$, $\gamma_t = 2(1+\theta_t)\eta$,$\beta = \frac{1}{d\sqrt{T}}$. The parameter b for Acc-SZOFW is set to 1 because in the algorithm it refers to the number of sample image for the stochastic setting, but in our configuration it refers only to a single image from which pixels are sampled.

Becuase of the limits of our personal devices, the computations were performed with Google Colab exploiting the GPU, but still we had to use a reduced number of iterations because of the time limit of the environment when executing a demanding code. Furthermore, to ease the computation and the time required by the *for* cycles, we exploited vectors of b_1, b_2 copies of the same image do perform parallel computations.

7 Results

7.1 MNIST

In the figure 1 there are the plots of the accuracy of the targeted model on the modified test set images and the distorsion. We can notice how the accuracy actually drops from the initial 99% and the maximum distorsion does not go over the boundary so we can say that our implementation is correct. In addition, we can see how the accuracy drops just to to ∼75% after 10 iterations for FWBB in both options, while it drops even to to ∼0.03% in just one iteration for Acc-ZOFW, so we can say that the authors of [2] actually developed an accelerated version. Our adjustments of the stochastic setting of Acc-SZOFW to SAP led to the results we expected, with Acc-SZOFW (Unige) in the middle between FWBB and Acc-ZOFW at 0.35% of accuracy, while Acc-SZOFW (Cooge) is in line with FWBB.

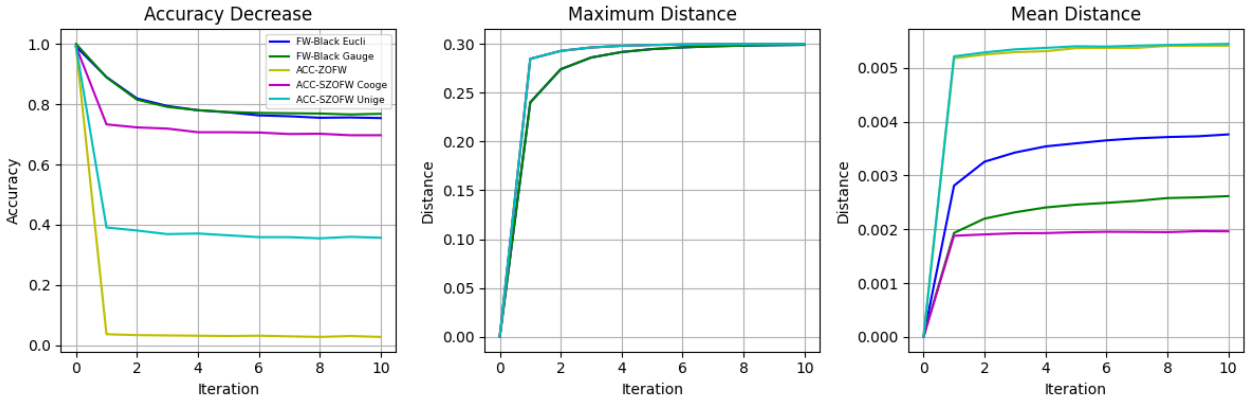

Figure 1: Performances on MNIST dataset

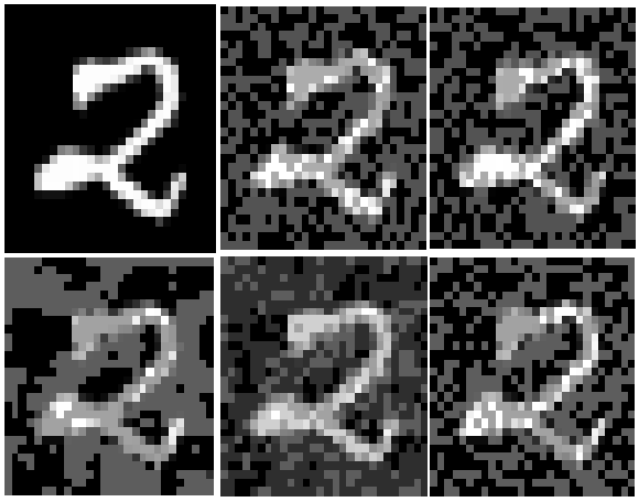

Figure 2: From left to right: the original image, FWBB (euclidean), FWBB (gaussian), Acc-ZOFW, Acc-SZOFW (cooge), Acc-SZOFW (unige)

Finally in figure 2 we show how the original image of the Mnist dataset looks like after it has been modified. A human observer even after the modification could still guess correctly the number represented in all cases, while the attacked DNN fails for the image modified by Acc-ZOFW predicting 5 instead of 2.

7.2 CIFAR10

We analyze the results of the CIFAR10 dataset in figure 3. Again the accuracy decreases and the distorsion remains in the boundary so the implementation remains correct. Furthermore, we can notice in the accuracy plot a pattern very similar to the previous one but this time it drops more for all the algorithms. It must be remarked that this time the accuracy started from 0.70% and that we only corrupted 100 images, whereas before they were 1000.

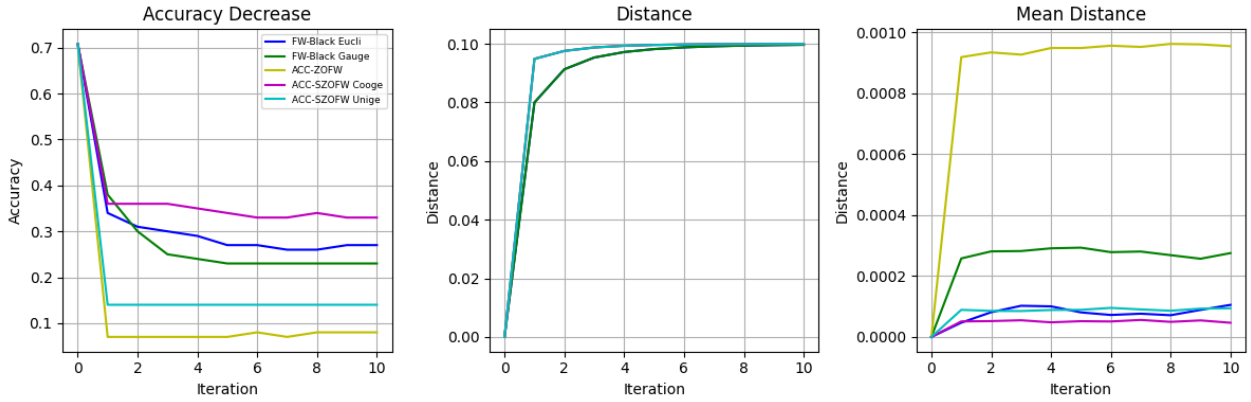

Figure 3: Performances on MNIST dataset

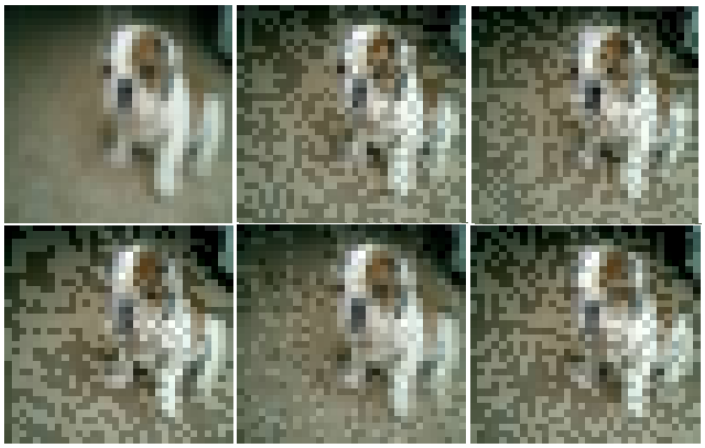

Figure 4: From left to right: the original image, FWBB (euclidean), FWBB (gaussian), Acc-ZOFW, Acc-SZOFW (cooge), Acc-SZOFW (unige)

In figure 4 we show the original image of the CIFAR10 dataset looks like after it has been modified. A human observer this time may have a harder time classifying correctly the image but will certainly recognize that the image has not been completely distorted. In this case, also partly due to the model starting from an accuracy of 0.70% the DNN assigned the wrong label (4:deer instead of 5:dog).

We report in table 1 the final results of the accuracy of the algorithms on the datasets.

8 Conclusions

We successfully implemented the assigned algorithms FWBB and Acc-ZOFW and they turned out to be satisfactory in their performances, affecting the accuracy of the target DNN without distorting the original input data too much. Among all the algorithms Acc-ZOFW turned out to be the best.

Dataset	FWBB (Euclidean)	FWBB (Gaussian)	Acc-ZOFW	Acc-SZOFW (Cooge)	Acc-SZOFW (Unige)
Mnist	0.75	0.76	0.03	0.69	0.35
CIFAR10	0.27	0.23	0.07	0.33	0.14

Table 1: Accuracy results

References

[1] Jinghui Chen et al. "A frank-wolfe framework for efficient and effective adversarial attacks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 3486–3494.

[2] Feihu Huang, Lue Tao, and Songcan Chen. "Accelerated stochastic gradient-free and projection-free methods". In: *International conference on machine learning*. PMLR. 2020, pp. 4519–4530.