Zeroth-Order Frank-Wolfe Optimization for Black-Box Adversarial Attacks

Marco Uderzo
marco.uderzo@studenti.unipd.it
Student ID: 2096998

Abstract

The goal of this project is to compare the behaviour and performance of two Zeroth-Order variants of the Frank-Wolfe Algorithm, aimed at solving constrained optimization problems with a better iteration complexity, expecially with respect to oracle queries. We take into consideration: Faster Zeroth-Order Conditional Gradient Sliding (FZCGS) (Gao et al., 2018) and Stochastic Gradient Free Frank Wolfe (SGFFW) (Sahu et al., 2019). The latter algorithm branches off into three slightly different ones, depending on the Stochastic Approximation Technique used, namely: classical Kiefer-Wolfowitz Stochastic Approximation (KWSA) (Kiefer and Wolfowitz, 1952), Random Directions Stochastic Approximation (RDSA) (Nesterov and Spokoiny, 2011; Duchi et al., 2015), and an Improvised RDSA (IRDSA). The theory behind these algorithms is presented, with an emphasis on proving that the performance are guaranteed. Then, the aforementioned algorithms are tested on a black-box adversarial attack on the MNIST dataset.

1. Introduction

The Frank-Wolfe algorithm, also known as the conditional gradient method, is an iterative optimization technique used for constrained convex optimization problems. It was proposed by Marguerite Frank and Philip Wolfe in 1956, and nowadays finds various applications in the field of machine learning. It approximates the objective function by a first-order Taylor approximation. The algorithm iteratively selects a direction that minimizes the linear approximation of the objective function within the feasible set C. This direction is then combined with the current solution in a convex combination, and the process is repeated until convergence.

In particular, the Frank-Wolfe algorithm excels in constrained optimization problems with a closed convex set C:

$$\min_{x \in C} f(x)$$

The problem formulation can vary widely; for example, Gao et al. deal with a variant tailored for finite-sum minimization problems, in which the component functions $f_i(x)$ are summed up as follows:

$$\min_{x \in \Omega} F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Sahu et al., on the other hand, in order to estimate the loss function f(x), deal with a variant that uses a stochastic zeroth-order oracle. The loss function is therefore defined as:

$$\min_{x \in C} f(x) = \min_{x \in C} \mathbb{E}_y[F(x, y)]$$

This paper addresses the use of the Frank-Wolfe algorithm for a particularly critical constrained optimization problem in Deep Learning, which is the problem of Adversarial Attacks. The objective of an adversarial attack is to find a small enough perturbation of the input able to make the neural network output the wrong prediction, while adhering to constraints inside of the convex set C. In our case, we use MNIST, so the goal is to find a non-trivial perturbation of the 28x28 black and white image of a hand-written digit. Moreover, as we will see later, we employ a zeroth-order variant of the Frank-Wolfe algorithm, since we don't have access to the full exact gradient.

1.1. Deterministic Frank-Wolfe Algorithm

In case first-order information is available in an optimization task, the deterministic version of the Frank-Wolfe algorithm can be a good choice, expecially when exact minimization is computationally expensive.

The exact minimization in the first formula in the introduction is approximated through an inexact minimization, where a vector v satisfies some conditions, while maintaining the same convergence rate.

When full, exact first-order information is available through an incremental first-order oracle (IFO), the Frank-Wolfe algorithm is basically described by the following two formulas:

$$v_t = \arg\min_{v \in \mathcal{C}} \langle h, \nabla f(x_t) \rangle$$
$$x_{t+1} = (1 - \gamma_{t+1}) x_t + \gamma_{t+1} v_t,$$

where

- $f(x_t)$ is the objective function we need to minimize;
- $\mathcal{C}$ is the convex set;
- $\langle \cdot, \cdot \rangle$ is the inner/dot product;
- v_t is the direction we need to take in order to minimize the linear approximation;
- x_t is the current iteration result;
- h is a vector in the same space as x_t;
- $\gamma_{t+1} = \frac{2}{t+2}$ is the step size.

1.2. Stochastic Frank-Wolfe Algorithm

In case first-order information is not available, and we can only work with zeroth-order information, the stochastic variant of the Frank-Wolfe algorithm can be a good choice.

By employing a Stochastic Zeroth-order Oracle (SZO), the deterministic objective function is substituted by a stochastic objective function $f(x_t, y_t)$, with y_t being a random variable. Therefore, the Stochastic Frank-Wolfe algorithm becomes:

$$v_t = \arg\min_{v \in \mathcal{C}} \langle h, \nabla f(x_t, y_t) \rangle$$
$$x_{t+1} = (1 - \gamma_{t+1}) x_t + \gamma_{t+1} v_t,$$

1.3. Zeroth-Order Gradient Estimation

When the gradient of a function is unavailable, it is possible to estimate it using function evaluations, by calculating the function values at selected points. More in detail, we can use the difference of the function value with respect to two random points to estimate the gradient. In our case, we employ the use of the coordinate-wise gradient estimator, as in the Gao et al. paper.

The coordinate-wise gradient estimator is defined as follows:

$$\hat{\nabla} f(\mathbf{x}) = \sum_{j=1}^{d} \frac{f(\mathbf{x} + \mu_j \mathbf{e}_j) - f(\mathbf{x} - \mu_j \mathbf{e}_j)}{2\mu_j} \mathbf{e}_j$$

where

- $\hat{\nabla} f(\mathbf{x})$ is the estimated gradient of the function f in $\mathbf{x}$
- d is the dimensionality of the optimization space
- $\mu_j > 0$ is a smoothing parameter
- $\mathbf{e}_j \in \mathbb{R}^d$ is the basis vector where only the j-th element is 1 and all others are 0.

2. Implemented Algorithms

This project involves the implementation of the following algorithms:

- **FZCGS**: Faster Zeroth-Order Conditional Gradient Sliding Method
- **SGFFW**: Stochastic Gradient-Free Frank-Wolfe with the following gradient approximation schemes:
  - **KWSA**: Kiefer-Wolfowitz stochastic approximation
  - **RDSA**: random directions stochastic approximation
  - **I-RDSA**: improvised random directions stochastic approximation

2.1. FZCGS: Faster Zeroth-Order Conditional Gradient Sliding Method

Gao et al. proposes a novel algorithm to optimize the following constrained finite-sum minimization problem:

$$\min_{x \in \Omega} F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

where

- $\Omega \subset \mathbb{R}^d$ is the closed convex feasible set
- $f_i(x)$ are the various n component functions, which are smooth and non-convex

FZCGS incorporates an acceleration technique from the non-convex Frank-Wolfe method. (?) It uses gradient estimation and Conditional Gradient Sliding to perform the updates. The convergence rate depends on the choice of the parameters. The pseudocode of the algorithm is presented below.

Algorithm 2 Faster Zeroth-Order Conditional Gradient Method (FZCGS)

Input: $\mathbf{x}_0, q > 0, \mu > 0, K > 0, \eta > 0, \gamma > 0, n$

for $k = 0, \cdots, K-1$ **do**
  if mod(k, q) = 0 **then**
    Sample S_1 without replacement to compute $\hat{\mathbf{v}}_k = \hat{\nabla} f_{S_1}(\mathbf{x}_k)$
  else
    Sample S_2 with replacement to compute $\hat{\mathbf{v}}_k = \frac{1}{|S_2|}\sum_{i \in S_2}[\hat{\nabla} f_i(\mathbf{x}_k) - \hat{\nabla} f_i(\mathbf{x}_{k-1}) + \hat{\mathbf{v}}_{k-1}]$
  end if
  $\mathbf{x}_{k+1} = \text{condg}(\hat{\mathbf{v}}_k, \mathbf{x}_k, \gamma_k, \eta_k)$
end for

Output: Randomly choose $\mathbf{x}_\alpha$ from $\{\mathbf{x}_k\}$ and return it

Algorithm 3 $\mathbf{u}^+ = \text{condg}(\mathbf{g}, \mathbf{u}, \gamma, \eta)$ (Qu et al., 2017)

$\mathbf{u}_1 = \mathbf{u}, t = 1$

$\mathbf{v}_t$ be an optimal solution for

$$V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x}\in\Omega}\langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x}\rangle$$

If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \le \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.

Set $\mathbf{u}_{t+1} = (1-\alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t)-\mathbf{g}, \mathbf{v}_t-\mathbf{u}_t\rangle}{\frac{1}{\gamma}\|\mathbf{v}_t-\mathbf{u}_t\|^2}\}$.

Set $t \leftarrow t+1$ and goto step 2.

The results of FZCGS from the paper are:

- When choosing the right set of parameters for FZCGS, the expected squared norm of the gradient estimation error converges as per the given rate
- The amortized function queries oracle complexity is $O\left(\frac{n^{1/2}d}{\epsilon}\right)$.
- The linear oracle complexity is $O\left(\frac{1}{\epsilon^2}\right)$.

In the paper, the experimental results demonstate the superiority of FZCGS over baseline methods in terms of onvergence speed and performance, also thanks to the acceleration technique that this algorithm employs.

2.2. Stochastic Gradient-Free Frank-Wolfe Algorithm

Sahu et al. propose a stochastic zeroth-order Frank-Wolfe algorithm for the following stochastic optimization problem, particularly significant in deep learning:

$$\min_{x\in C} f(x) = \min_{x\in C} \mathbb{E}_y[F(x;y)]$$

where $C \in \mathbb{R}^d$ is a closed convex set.

To solve this kind of problem, one can employ projection and projection-free methods. SGFFW uses a zeroth-order oracle while focusing on a projection-free stochastic variant of Frank-Wolfe. It uses gradient approximation schemes and addresses challenges such as non-smoothness and variance. Using an averaging trick to ensure the stability of the algorithm, the updates are the following:

$$\begin{aligned} d_t &= (1-\rho_t)\, d_{t-1} + \rho_t g\,(x_t, y_t) \\ v_t &= \arg\min_{s\in C} \langle s, d_t\rangle \\ x_{t+1} &= (1-\gamma_{t+1})\, x_t + \gamma_{t+1} v_t \end{aligned}$$

Employing a gradient approximation techniques or another influences the algorithm's dimension dependence and the computational cost, also in terms of query number and time.

- Kiefer-Wolfowitz Stochastic Approximation (KWSA):

$$\mathbf{g}\,(\mathbf{x}_t;\mathbf{y}) = \sum_{i=1}^{d} \frac{F\,(\mathbf{x}_t + c_t\mathbf{e}_i;\mathbf{y}) - F\,(\mathbf{x}_t;\mathbf{y})}{c_t}\mathbf{e}_i$$

- Random Direction Stochastic Approximation (RDSA):
  Sample $\mathbf{z}_t \sim \mathcal{N}\,(0, \mathbf{I}_d)$,

$$\mathbf{g}\,(\mathbf{x}_t;\mathbf{y},\mathbf{z}_t) = \frac{F\,(\mathbf{x}_t + c_t\mathbf{z}_t;\mathbf{y}) - F\,(\mathbf{x}_t;\mathbf{y})}{c_t}\mathbf{z}_t$$

- Improvised Random Direction Stochastic Approximation (I-RDSA):
  Sample$\{\mathbf{z}_{i,t}\}_{i=1}^{m} \sim \mathcal{N}\,(0, \mathbf{I}_d)$

$$\mathbf{g}\,(\mathbf{x}_t;\mathbf{y},\mathbf{z}_t) = \frac{1}{m}\sum_{i=1}^{m} \frac{F\,(\mathbf{x}_t + c_t\mathbf{z}_{i,t};\mathbf{y}) - F\,(\mathbf{x}_t;\mathbf{y})}{c_t}\mathbf{z}_{i,t}$$

Below, the pseudocode of the algorithm is presented:

Algorithm 2 Stochastic Gradient Free Frank Wolfe

Require: Input, Loss Function $F(x)$, Convex Set $\mathcal{C}$, number of directions m, sequences $\gamma_t = \frac{2}{t+8}$,
$(\rho_t, c_t)_{RDSA} = \left(\frac{4}{d^{1/3}(t+8)^{2/3}}, \frac{2}{d^{3/2}(t+8)^{1/3}}\right)$
$(\rho_t, c_t)_{I-RDSA} = \left(\frac{4}{\left(1+\frac{d}{m}\right)^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}}\right)$
$(\rho_t, c_t)_{KWSA} = \left(\frac{4}{(t+8)^{2/3}}, \frac{2}{d^{1/2}(t+8)^{1/3}}\right)$.

Output: x_T or $\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{x}_t$.

1: Initialize $\mathrm{x}_0 \in \mathcal{C}$
2: **for** $t = 0, 2, \ldots, T-1$ **do**
3: Compute
KWSA:
$\mathrm{g}(\mathrm{x}_t; \mathrm{y}) = \sum_{i=1}^{d} \frac{F(\mathbf{x}_t + c_t \mathbf{e}_i; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{e}_i$
RDSA: Sample $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathrm{g}(\mathrm{x}_t; \mathrm{y}, \mathrm{z}_t) = \frac{F(\mathbf{x}_t + c_t \mathbf{z}_t; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{z}_t$
I-RDSA: Sample $\{\mathbf{z}_{i,t}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathrm{g}(\mathrm{x}_t; \mathrm{y}, \mathrm{z}_t) = \frac{1}{m}\sum_{i=1}^{m} \frac{F(\mathbf{x}_t + c_t \mathbf{z}_{t,i}; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{z}_{i,t}$
4: Compute $\mathbf{d}_t = (1 - \rho_t)\, \mathbf{d}_{t-1} + \rho_t \mathrm{g}\left(\mathrm{x}_t, \mathrm{y}_t\right)$
5: Compute $\mathbf{v}_t = \mathrm{argmin}_{\mathbf{s} \in \mathcal{C}} \langle \mathbf{s}, \mathbf{d}_t \rangle$),
6: Compute $\mathrm{x}_{t+1} = (1 - \gamma_t)\, \mathrm{x}_t + \gamma_t \mathbf{v}_t$.
7: **end for**

3. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [1]. Where appropriate, include the name(s) of editors of referenced books.

3.1. Illustrations, graphs, and photographs

All graphics should be centered. Please ensure that any point you wish to make is resolvable in a printed copy of the paper. Resize fonts in figures to match the font in the body text, and choose line widths which render effectively in print. Many readers (and reviewers), even of an electronic copy, will choose to print your paper in order to read it. You cannot insist that they do otherwise, and therefore must not assume that they can zoom in to see tiny details on a graphic.

When placing figures in LaTeX, it's almost always best to use `\includegraphics`, and to specify the figure width as a multiple of the line width as in the example below

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]
                {myfile.eps}
```

References

[1] Authors. The frobnicatable foo filter, 2014. Face and Gesture submission ID 324. Supplied as additional material `fg324.pdf`.