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

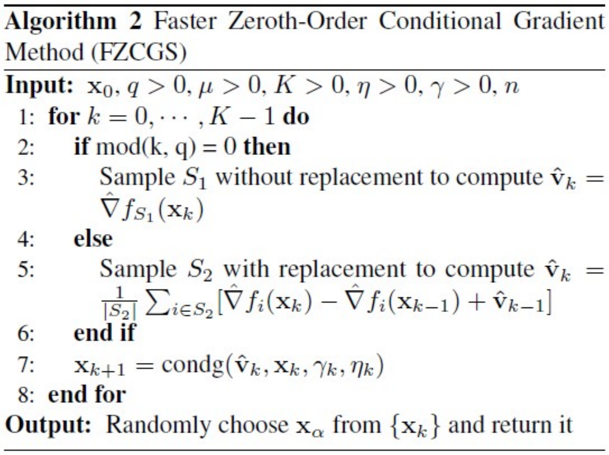
Gao et al. proposes a novel algorithm to optimize the following constrained finite-sum minimization problem:

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

where

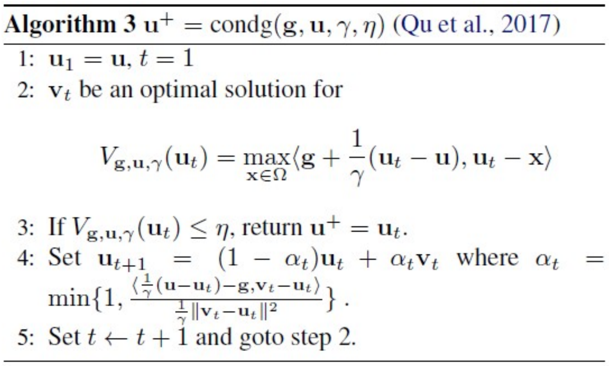
- $\Omega \subset \mathbb{R}^d$ is the closed convex feasible set
- $f_i(x)$ are the various n component functions, which are smooth and non-convex

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
    $\mathbf{x}_{k+1} = \mathrm{condg}(\hat{\mathbf{v}}_k, \mathbf{x}_k, \gamma_k, \eta_k)$
end for
Output: Randomly choose $\mathbf{x}_\alpha$ from $\{\mathbf{x}_k\}$ and return it
```



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

```
1: $\mathbf{u}_1 = \mathbf{u}, t = 1$
2: $\mathbf{v}_t$ be an optimal solution for
       $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x} \in \Omega} \langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x} \rangle$
3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \leq \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.
4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t) - \mathbf{g}, \mathbf{v}_t - \mathbf{u}_t \rangle}{\frac{1}{\gamma}\|\mathbf{v}_t - \mathbf{u}_t\|^2}\}$.
5: 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 convergence 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):

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

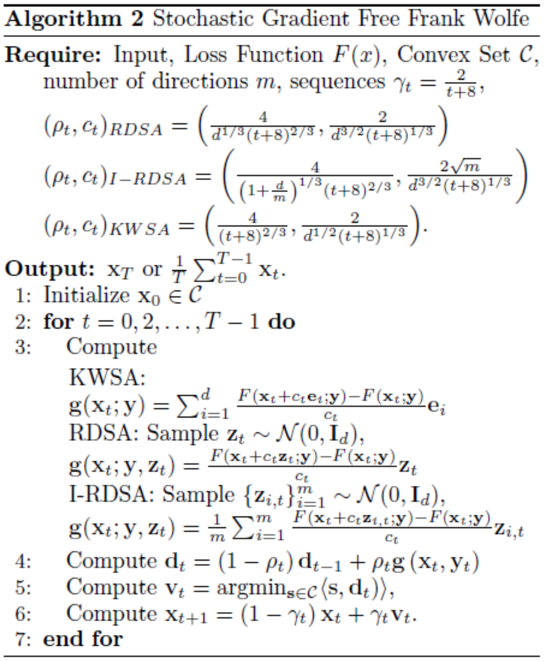
- Random Direction Stochastic Approximation (RDSA):
  Sample $\mathbf{z}_t \sim \mathcal{N}\,(0, \mathbf{I}_d)$,

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

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

2.2.1 Primal Gap

Sahu et al. state the main results involving the different gradient approximation schemes for the primal (suboptimality) gap, which provide a characterization of

$$E\left[f\left(x_t\right)-f\left(x^*\right)\right]$$

.

Given the sequence $\gamma_t = \frac{2}{t+8}$:

- RDSA gradient approximation scheme:

$$E\left[f\left(x_t\right)-f\left(x^*\right)\right]=O\left(d^{1/3}(t+9)^{1/3}\right)$$

- I-RDSA gradient approximation scheme:

$$E\left[f\left(x_t\right)-f\left(x^*\right)\right]=O\left(\frac{d}{m^{1/3}}(t+9)^{1/3}\right)$$

- KWSA gradient approximation scheme:

$$E\left[f\left(x_t\right)-f\left(x^*\right)\right]=O\left(\frac{1}{(t+9)^{1/3}}\right)$$

The dimension dependence of the primal gap is quantified to be $d^{1/3}$. At the same time, the dependence on iterations $O\left(T^{-1/3}\right)$, matches that of the stochastic Frank-Wolfe which has access to first-order information. The improvement of the rates for I-RDSA and KWSA are at the cost of extra directional derivatives at each iteration. The number of queries to the SZO to obtain a primal gap of ϵ is given by $O\left(\frac{d}{\epsilon^3}\right)$, where the dimension dependence is consistent with zeroth-order schemes and cannot be improved on.

2.2.2 Dual Gap

The paper also quantifies the dual gap with the same gradient approximation schemes

- RDSA gradient approximation scheme:

$$\begin{aligned} E\left[\min_{t=0,\ldots,T-1} G\left(x_t\right)\right] \leq & \frac{7\left(F\left(x_0\right)-F\left(x^*\right)\right)}{2T} \\ & + \frac{LR^2 \ln(T+7)}{T} \\ & + \frac{Q_0 + R\sqrt{2Q}}{2T^{2/3}} \end{aligned}$$

- I-RDSA gradient approximation scheme:

$$\begin{aligned} E\left[\min_{t=0,\ldots,T-1} G\left(x_t\right)\right] \leq & \frac{7\left(F\left(x_0\right)-F\left(x^*\right)\right)}{2T} \\ & + \frac{LR^2 \ln(T+7)}{T} \\ & + \frac{Q_{ir} + R\sqrt{2Q_{ir}}}{2T^{2/3}} \end{aligned}$$

- KWSA gradient approximation scheme:

$$\begin{aligned} E\left[\min_{t=0,\ldots,T-1} G\left(x_t\right)\right] \leq & \frac{7\left(F\left(x_0\right)-F\left(x^*\right)\right)}{2T} \\ & + \frac{LR^2 \ln(T+7)}{T} \\ & + \frac{Q_{kw} + R\sqrt{2Q_{kw}}}{2T^{2/3}} \end{aligned}$$

The dimension dependence of the Frank-Wolfe duality gap is quantified to be $d^{1/3}$. At the same time, the dependence on iterations, $O\left(T^{-1/3}\right)$, matches that of the primal gap and hence follows that the number of queries to the SZO to obtain a Frank-Wolfe duality gap of ϵ is given by $O\left(\frac{d}{\epsilon^3}\right)$. In particular, it is also asserted that the initial conditions are forgotten as $O(1/T)$.

3. Frank-Wolfe for Adversarial Attacks on MNIST

3.1. Adversarial Attacks

An Adversarial Attacks is a deliberate manipulation of input data with the intention of causing a machine learning model to make a mistake or produce incorrect outputs. The goal of an adversarial attack is to perturb the input in a way that is not easily noticeable to a human observer, but can mislead the model into making errors. In particular, when it comes to black-box attacks, the attacker has limited or no knowledge of the model's internal structure and parameters. These attacks can be performed on deep neural networks models; in our case, the model is MNIST, designed for the classification of handwritten digits. The MNIST dataset consists of 28x28 pixel grayscale images of handwritten digits (0 through 9). The objective of our experiment is to compare the optimization of an adversarial attack using zeroth-order Frank-Wolfe algorithms in a black-box setting. The task is to find a non-trivial adversarial perturbation such that the DNN model makes the incorrect prediction.

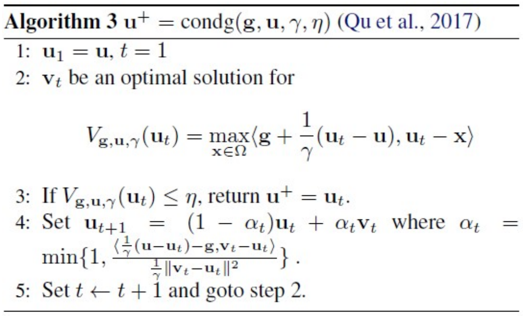
3.2. Methods

We follow the setup (Liu et al., 2019) with the nn-carlini pre-trained Deep Neural Network for the MNIST dataset; this base repository is also used by Gao et al. for their experiments, so starting from it as a base for the implementation of the optimization algorithms was deemed to be a good way to set up our experiments as well. The base repository, as per Gao et al., can be found at the following link.

3.2.1 Implementation Differences from Reference Papers

For time and computational constraints, there are some implementative differences between the theoretical papers and our actual implementation.

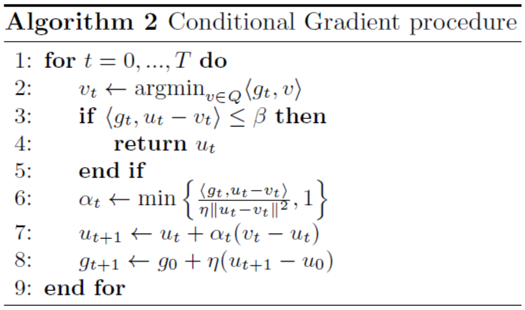
- In FZCGS, the `condg` function from Gao et al. has been swapped for the same conditional gradient procedure from Lobanov et al., because it was failing to learn. Though, the pseudocode implementation described by Lobanov et al. works correctly, and should provide the same result as the Gao et al. pseudocode implementation.



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

```
1: $\mathbf{u}_1 = \mathbf{u}, t = 1$
2: $\mathbf{v}_t$ be an optimal solution for
       $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x}\in\Omega}\langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x}\rangle$
3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \le \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.
4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t)-\mathbf{g}, \mathbf{v}_t-\mathbf{u}_t\rangle}{\frac{1}{\gamma}\|\mathbf{v}_t-\mathbf{u}_t\|^2}\}$.
5: Set $t \leftarrow t + 1$ and goto step 2.
```

Conditional Gradient procedure (Gao et al.)



Algorithm 2 Conditional Gradient procedure

```
for $t = 0, ..., T$ do
    $v_t \leftarrow \mathrm{argmin}_{v\in Q}\langle g_t, v\rangle$
    if $\langle g_t, u_t - v_t\rangle \le \beta$ then
        return $u_t$
    end if
    $\alpha_t \leftarrow \min\left\{\frac{\langle g_t, u_t - v_t\rangle}{\eta\|u_t - v_t\|^2}, 1\right\}$
    $u_{t+1} \leftarrow u_t + \alpha_t(v_t - u_t)$
    $g_{t+1} \leftarrow g_0 + \eta(u_{t+1} - u_0)$
end for
```

Conditional Gradient procedure (Lobanov et al.)

- In the `condg` (conditional gradient) step for FZCGS, we added an additional stopping condition on the size of alpha. This was done because the conditional gradient step, if not limited on iteration number, the stopping condition would not be reached even after a great amount of iterations. This, together with the extremely large query count that FZCGS requires, would have made infeasible the optimization process, because of time constraints. The newly added stopping condition speeds up the conditional gradient function running time, whilst causing negligible deviations from the original result.

3.2.2 Feasible Set

The objective function used is the same from the base repository. The algorithms were tested on the feasible set $\|\delta\|_\infty \le s$, which is the same constraint that Gao et al. references in section 4.3 for the formulation of the problem of generation of adversarial examples.

4. Results

The general parameters (non algorithm-specific) for the setup are presented in the table below.

Parameter	Value	Description
nStage	200	Iterations
targetlabel	4	Target digit
nFunc	10	Imgs to attack simult.
const	3	Weight on attack loss
rvdist	UnitSphere	Random perturb. distr.

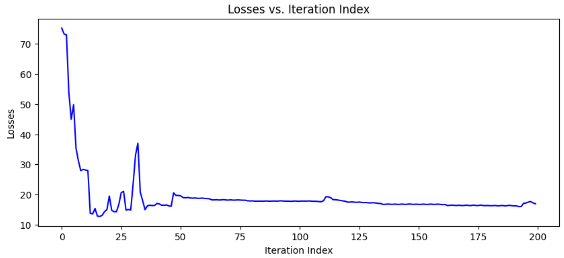
Table 1. General Setup Parameters

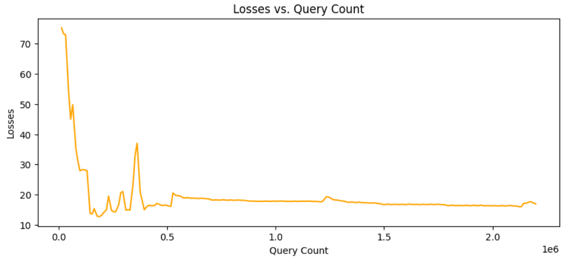
4.1. FZCGS

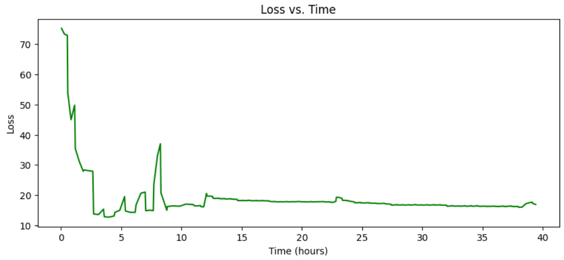
The FZCGS algorithm presented in Gao et al. is evaluated with respect to loss against iteration count, query count and running time, as the below graphs show:

Parameter	Value	Description
nStage	200	Iterations
q	3	Batch size S2
K	0.1	K parameter
L	50	Lipschitz constant
mu	0.11	mu parameter in Gao
gamma	0.01	gamma parameter in Gao
eta	0.1	Tolerance parameter in Gao
s	4	Value for $\|\delta\|_\infty \leq s$

Table 2. Parameters of the FZCGS Algorithm

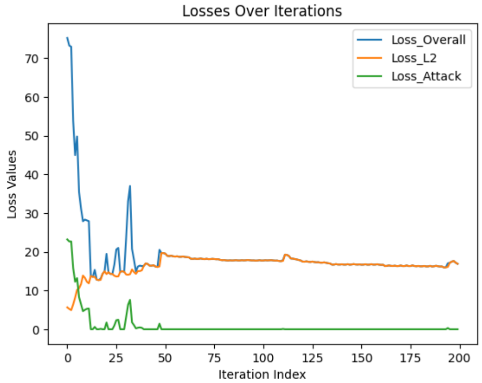






It is clear from the graphs that 200 iterations are enough for FZCGS to converge to a good loss, though, since it uses multiple stochastic examples at each epoch for each direction analyzed, performances considering query count and time are where FZCGS presents criticalities. In fact, as we will discuss later, it is the worst algorithm in terms of running time between the ones considered in this study.

Moreover, we can see that after around 50 iterations, the algorithm stops improving in regards of the overall loss. This has to be attributed to the `Loss Attack` dropping to zero, and this makes FZCGS converge to a higher loss compared to other SGFFW variants, as we show in the next section. The following graph shows this behaviour:



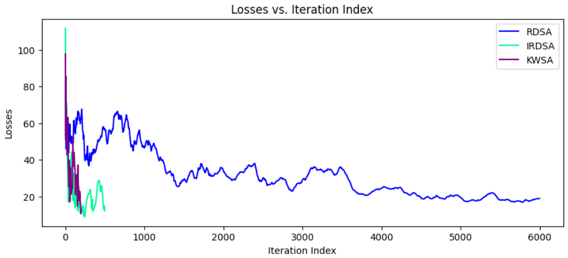
Interestingly, FZCGS reaches the best loss in the whole run at the 16th iteration (out of 200), though we regard it as an outlier value, since the loss for the rest of the run stays over the best loss.

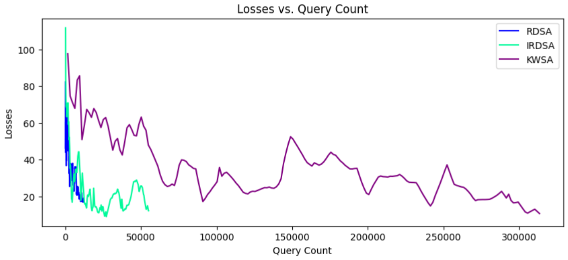
4.2. SGFFW

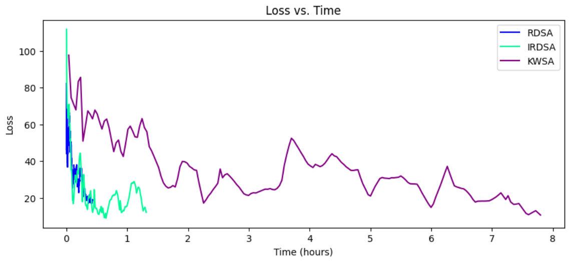
The SGFFW algorithm from Sahu et al. was evaluated with the three gradient approximation schemes (RDSA, I-RDSA, KWSA). Below, the comparison between these variants is presented, comparing the overall losses against iteration count, query count and running time.

Parameter	Value	Description
nStage	200	Iterations
gamma	$\frac{2}{t+8}$	gamma parameter
m	50	Number of random vectors
s (RDSA)	8	Value for $\|\delta\|_\infty \leq s$
s (I-RDSA)	4	
s (I-RDSA)	4	

Table 3. Parameters of the SGFFW Algorithm







SGFFW has different pretty different behaviours when the three gradient approximation schemes are compared.

The RDSA gradient approximation scheme is the fastest between all other in terms of time, but requires many more iterations to reach a comparable loss value. Though, the query count of RDSA is very limited, which is beneficial for the running time.

I-RDSA requires less iterations to reach a similar loss to RDSA, at the expense to a bigger query count and a larger running time.

Between gradient approximation schemes for SGFFW, KWSA is the worst in terms of time, requiring a very considerable amount of time to reach a good loss, while performing the least amount of iterations. Other performed tests show that increasing iterations does not yield better results.

Considering iterations only, RDSA requires the most of them and therefore is the worst algorithm, but since the query count requirement for each iteration is very low, it is the fastest.

Since this study focuses more on query count performance, whereas iteration count is also taken into account in other papers, we can say that overall I-RDSA performs best between the three approximation schemes for SGFFW, providing low query count, low iteration requirements and a reasonable running time to converge.

4.3. All Algorithms Compared

In this section, we are going to compare all the algorithms and determine which performing best, taking into account the best loss achieved, the last loss computed, the iteration count, the query count and running time, measured in hours.

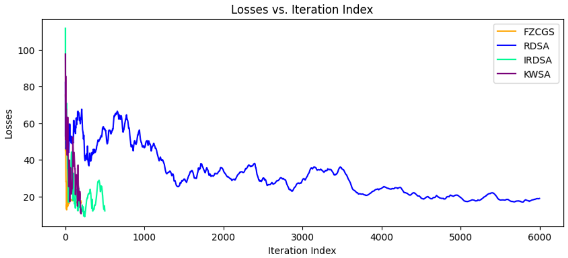
Notably, we see that I-RDSA converged to the best loss, followed by KWSA and FZCGS. RDSA, while being the fastest, converged to the worst loss between all algorithms.

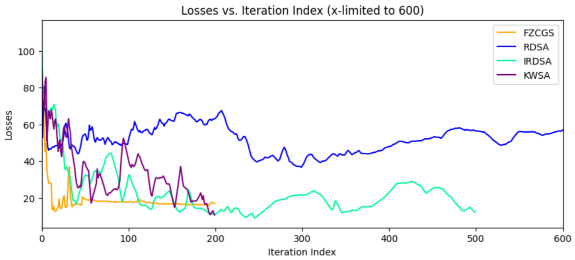
Below, the result graphs are presented. Each graph is also replotted while limiting the x-axis, to better appreciate the behaviour of each algorithm.

Algorithm	Metric	Value
FZCGS	Best Loss	12.689 @ 16th it.
	Last Loss	16.9
	Iteration Count	200
	Query Count	2198768
	RunTime (Hrs)	39.4
SGFFW-RDSA	Best Loss	16.9 @ 5788th it.
	Last Loss	18.922
	Iteration Count	6000
	Query Count	12000
	RunTime (Hrs)	0.43
SGFFW-I-RDSA	Best Loss	8.96 @ 245th it.
	Last Loss	12.156
	Iteration Count	500
	Query Count	55000
	RunTime (Hrs)	1.3
SGFFW-KWSA	Best Loss	10.647 @ 199th it.
	Last Loss	10.647
	Iteration Count	200
	Query Count	313600
	RunTime (Hrs)	7.7

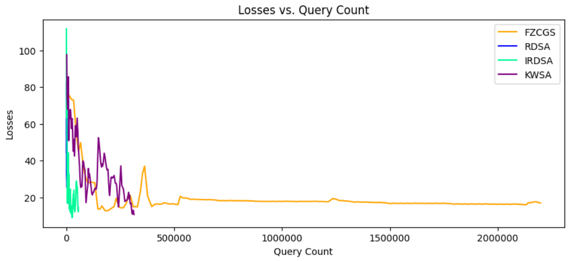
Table 4. Overall results and statistics compared

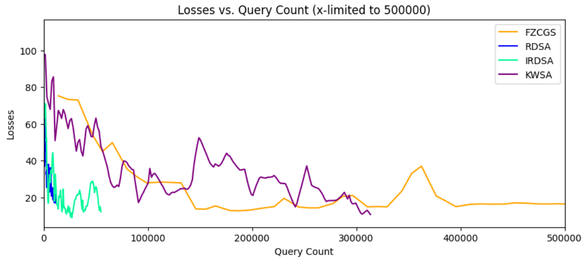
Losses vs Iterations (All)



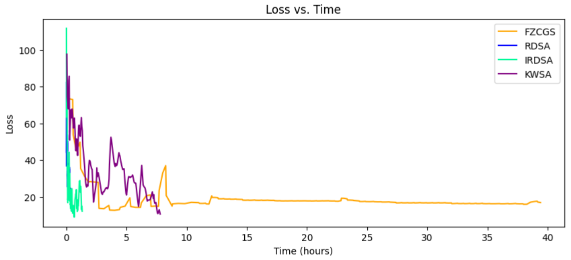


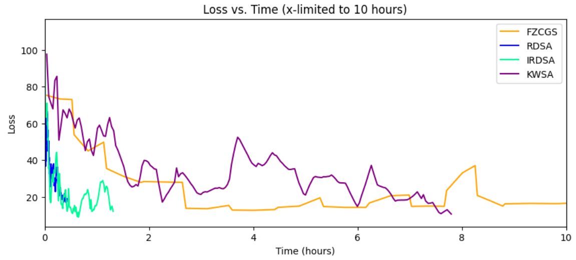
Losses vs Query Count (All)





Losses vs Running Time in Hours (All)





5. Conclusion

Overall, there is no overall best method, and each one has its own strengths and weaknesses.

From our study, FZCGS and SGFFW with KWSA had the best performance when considering iterations; however, when compared to all of the SGFFW variants, they are the worst in terms of query count and running time, making them less favourable.

SGFFW with RDSA is the fastest for running time, but having the worst iterations count required to converge and reaching the highest loss among all algorithms considered.

Nonetheless, SGFFW with I-RDSA gradient approximation scheme showed a remarkable compromise between running time, iteration count and query count, making it the 'best' overall method for zeroth-order Frank-Wolfe optimization in the setting of adversarial attacks against the MNIST dataset.

References