Zeroth Order Frank-Wolfe: a comparison

Laura Iacovissi, Federico Matteo

University of Padova
Optimization for Data Science

July 11, 2020

1 Introduction

1.1 Stochastic Optimization

In this project we will apply stochastic optimization methods that solve the problem

$$\min_{x \in \mathcal{C}} f(\boldsymbol{x}) = \min_{x \in \mathcal{C}} \mathbb{E}_{\boldsymbol{y} \sim \mathcal{P}}[F(\boldsymbol{x}; \boldsymbol{y})], \tag{1}$$

where $\mathcal{C} \in \mathbb{R}^d$ is a closed convex set, $f(\cdot)$ is the expected loss function, possibly non-convex, F the random loss function ($\boldsymbol{x}$: deterministic component; $\boldsymbol{y}$ random component) defined on a sample space Ω with a probability distribution $\mathcal{P}$.

We consider the output reached by using the following definition of ε-precision as the solution of the problem.

Definition 1.1.1. Let $\hat{\boldsymbol{x}} \in X$ be an output of an algorithm solving (1) and $\varepsilon > 0$ a target accuracy. Then, if f is convex, we have that $\hat{\boldsymbol{x}}$ is called an ε-optimal point of problem (1) if $\mathbb{E}[f(\hat{\boldsymbol{x}})] - f(\boldsymbol{x}^*) \leq \varepsilon$, where $\boldsymbol{x}^*$ denotes an optimal solution of the problem.

The problem defined in (1) is of particular interest for data science applications, since it has been widely used in the context of deep learning and in solving both classification and regression problems on high dimensional datasets (namely expected risk minimization).

In the optimization framework, the problem (1) can be solved by applying algorithms that require a projection step or using projection free methods. In the latter case it is also possible to choose between various kinds of oracles, e.g. adopting first order oracles based on gradient computations or zeroth order oracles defined by function queries.

In this project, we aim to first introduce a stochastic version of the Frank-Wolfe algorithm, a projection free method, with access to various types of zeroth order oracles for the gradient. Then, we show the efficacy of the proposed algorithms with experiments on multiple datasets.

1.2 Frank-Wolfe Algorithm

The Frank-Wolfe method (aka conditional gradient method or reduced gradient method) is an iterative first-order optimization algorithm used for solving quadratic programming problems with linear constraints, when exact first order information is available. In this case, we assume to have

access to an incremental first order oracle (IFO), yielding to a deterministic Frank-Wolfe method, involving the following steps:

$$\boldsymbol{v}_t = \arg\min_{\boldsymbol{v}\in\mathcal{C}}(\nabla f(\boldsymbol{x}_t)\cdot\boldsymbol{v}), \tag{2}$$

$$\boldsymbol{x}_{t+1} = (1-\gamma_{t+1})\boldsymbol{x}_t + \gamma_{t+1}\boldsymbol{v}_t, \tag{3}$$

with γ_t suitable learning rate. Hence, a linear minimization oracle (LMO) is queried at every epoch.

The exact minimization in (2) is a linear program when $\mathcal{C}$ is given by linear constraints (i.e. a polytope) and can be performed efficiently without much computational overload, in particular if it is compared to a projection step.

As already said in the Introduction, we will not consider an IFO, but Zeroth Order Oracles (ZOO).

1.3 Zeroth Order Oracles

Derivative free optimization or zeroth order optimization is motivated by settings where the analytical form of the function is not available or when the gradient evaluation is computationally prohibitive, which is the experimental setup of the problems investigated in this work.

It is important to highlight that in the context of the optimization problem posed in (1), we assume that we have access to a stochastic zeroth order oracle (SZO) for the gradient and that on querying a SZO at the iterate $\boldsymbol{x}_t$, it yields an unbiased estimate of the loss function $f(\cdot)$ in the form of $F(\boldsymbol{x}_t;\boldsymbol{y}_t)$.

The main aspect of zeroth order optimization consists of gradient approximation schemes from appropriately sampled values of the objective function. In the literature, there exist only few zeroth order gradient approximation schemes, which are described and applied in the following.

1.3.1 The Kiefer-Wolfowitz stochastic approximation

The Kiefer-Wolfowitz stochastic approximation (KWSA) for the gradient queries the loss function along the canonical basis vectors. Formally, the gradient estimate can be expressed as

$$\boldsymbol{g}(\boldsymbol{x}_t;\boldsymbol{y}) = \sum_{i=1}^{d}\frac{F(\boldsymbol{x}_t + c_t\mathbf{e}_i;\boldsymbol{y}) - F(\boldsymbol{x}_t;\boldsymbol{y})}{c_t}\mathbf{e}_i, \tag{4}$$

with c_t time-decaying sequence, F with finite variance. Notice that KWSA requires d ZOO calls at each evaluation of the gradient, which can be costly for high dimensional applications.

1.3.2 The random directions gradient estimator

To avoid querying the objective loss function d times, random directions based gradient estimator (RDSA) have been proposed in the literature. The RDSA estimates the directional derivative along a randomly sampled direction from an appropriate probability distribution. Formally, it is defined as

$$\boldsymbol{g}(\boldsymbol{x}_t;\boldsymbol{y},\boldsymbol{z}_t) = \frac{F(\boldsymbol{x}_t + c_t\boldsymbol{z}_t;\boldsymbol{y}) - F(\boldsymbol{x}_t;\boldsymbol{y})}{c_t}\boldsymbol{z}_t, \tag{5}$$

where $\boldsymbol{z}_t \in \mathbb{R}^d$ is a random vector sampled from a probability distribution such that $\mathbb{E}[\boldsymbol{z}_t\boldsymbol{z}_t^\top] = \mathbf{I}_d$ and c_t is a (carefully chosen) time-decaying sequence. We will use as random distribution the standard normal one, $\mathcal{N}(\mathbf{0},\mathbf{I}_d)$.

We have that when $c_t \to 0$, both the gradient estimators in (4) and (5) are unbiased estimators of the gradient of the expected loss function, i.e. $\nabla f(\boldsymbol{x}_t)$.

1.3.3 The improvised random directions gradient estimator

In addition to the KWSA scheme and the RDSA scheme, as defined in (4) and (5) respectively, it is also possible to employ an improvised random directions gradient estimator (I-RDSA) by sampling m directions at each time followed by averaging, i.e. $\{\boldsymbol{z}_{i,t}\}_{i=1}^{m} \mid \boldsymbol{z}_{i,t} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ or any other distribution respecting the conditions stated for RDSA, for which we have

$$\boldsymbol{g}_m(\boldsymbol{x}_t; \boldsymbol{y}, \boldsymbol{z}_{i,t}) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{F(\boldsymbol{x}_t + c_t \boldsymbol{z}_{i,t}; \boldsymbol{y}) - F(\boldsymbol{x}_t; \boldsymbol{y})}{c_t} \boldsymbol{z}_{i,t} \right). \tag{6}$$

The above gradient approximation uses one data point while utilizing m directional derivatives, yielding to improvements in the dimension dependence of the primal gap, but at the cost of m calls to the ZOO, as will be shown later.

2 Zeroth Order Stochastic Methods

In this section we first introduce the mathematical assumptions required for the analysis and then we formally introduce the implemented algorithms.

Assumption 1. In problem (1) the set $\mathcal{C}$ is bounded with finite diameter R.

Assumption 2. F is convex and Lipschitz continuous with $\sqrt{\mathbb{E}\left[\|\nabla_x F(\boldsymbol{x}; \cdot)\|^2 \right]} \leq L_1$ for all $\boldsymbol{x} \in \mathcal{C}$.

Assumption 3. The expected function f is convex. Moreover, its gradient ∇f is L-Lipschitz continuous over the set $\mathcal{C}$, i.e. for all $x, y \in \mathcal{C}$

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{7}$$

Assumption 4. The $\boldsymbol{z}_t$'s are drawn from a distribution μ such that $M(\mu) = \mathbb{E}\left[\left\|\boldsymbol{z}_t^6\right\| \right]$ is finite, and for any vector $\boldsymbol{g} \in \mathbb{R}^d$, there exists a function $s(d) : \mathbb{N} \mapsto \mathbb{R}_+$ such that,

$$\mathbb{E}\left[\|(\boldsymbol{g} \cdot \boldsymbol{z}_t)\boldsymbol{z}_t\|^2 \right] \leq s(d) \|\boldsymbol{g}\|^2.$$

Assumption 5. The unbiased gradient estimates $\nabla F(\boldsymbol{x}; \boldsymbol{y})$ of $\nabla f(\boldsymbol{x})$, i.e. $\mathbb{E}_{\boldsymbol{y} \sim \mathcal{P}}[\nabla F(\boldsymbol{x}; \boldsymbol{y})] = \nabla f(\boldsymbol{x})$ satisfy

$$\mathbb{E}\left[\|\nabla F(\boldsymbol{x}; \boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^2 \right] \leq \sigma^2. \tag{8}$$

Assumptions 1-3 and 5 are standard in the context of stochastic optimization. Assumption 4 provides for the requisite moment conditions for the sampling distribution of the directions utilized for finding directional derivatives so as to be able to derive concentration bounds.

As already stated, we will consider μ to be $\mathcal{N}(\boldsymbol{0}, \mathbf{I}_d)$, so that $M(\mu) = d(d+2)(d+4) \approx d^3$ and $s(d) = d$.

2.1 Deterministic Zeroth Order Frank-Wolfe

Problem (1) can be re-stared as a deterministic one by

$$\min_{\boldsymbol{x} \in \mathcal{C}} F(\boldsymbol{x}). \tag{9}$$

Algorithm 1 Deterministic Zeroth Order Frank Wolfe algorithm.

Require: Input, Loss Function $F(x)$, Convex Set $\mathcal{C}$, Lipschitz constant for the gradients L, Parameters γ_t, c_t.
Output: $\boldsymbol{x}_T$ or $\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t$
Initialize $\boldsymbol{x}_0 \in \mathcal{C}$
for $t = 1, 2, \ldots, T$ **do**
    Compute $\boldsymbol{g}(\boldsymbol{x}_t)$ with KWSA as in (10)
    Compute $\boldsymbol{v}_t = \arg\min_{s\in\mathcal{C}}(\boldsymbol{s}\cdot\boldsymbol{g}(\boldsymbol{x_t}))$
    Set $\boldsymbol{x}_{t+1} = (1-\gamma_t)\boldsymbol{x}_t + \gamma_t\boldsymbol{v}_t$

It is possible to underline the equivalence of a typical zeroth order Frank-Wolfe framework to an inexact classical Frank-Wolfe optimization by considering the KWSA as

$$\boldsymbol{g}(\boldsymbol{x}_t) = \sum_{i=1}^{d}\frac{F(\boldsymbol{x}_t + c_t\mathbf{e}_i) - F(\boldsymbol{x}_t)}{c_t}\mathbf{e}_i \tag{10}$$

$$= \nabla F(\boldsymbol{x}_t) + \sum_{i=1}^{d}\frac{c_t}{2}\Big(\mathbf{e}_i\cdot\nabla^2 F(\boldsymbol{x}_t + \lambda_t c_t\mathbf{e}_i)\mathbf{e}_i\Big)\mathbf{e}_i \tag{11}$$

where $\lambda \in [0,1]$. The linear optimization step with the current gradient approximation reduces to:

$$\boldsymbol{v}\cdot\boldsymbol{g}(\boldsymbol{x}_t) = \boldsymbol{v}\cdot\nabla F(\boldsymbol{x}_t) + \frac{c_t}{2}\sum_{i=1}^{d}\Big(\mathbf{e}_i\cdot\nabla^2 F(\boldsymbol{x}_t + \lambda_t c_t\mathbf{e}_i)\mathbf{e}_i\Big)(\boldsymbol{v}\cdot\mathbf{e}_i) \tag{12}$$

$$\Rightarrow \min_{s\in\mathcal{C}}\boldsymbol{v}\cdot\boldsymbol{g}(\boldsymbol{x}_t) \leq \min_{s\in\mathcal{C}}\Big(s\cdot\nabla F(\boldsymbol{x}_t\Big) + \frac{c_t LRd}{2}. \tag{13}$$

The Deterministic Zeroth Order Frank-Wolfe is described in Algorithm 1.

Theorem 2.1.1. *If $c_t = \frac{\gamma_t}{d}$ and $\gamma_t = \frac{2}{t+1}$ for Algorithm 1 we obtain the bound*

$$F(\boldsymbol{x}_T) - F(\boldsymbol{x}^*) = \frac{Q_{ns}}{t+2}, \tag{14}$$

where $Q_{ns} = \max\{2(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)),\, 4LR^2\}$.

Notice that the γ_t parameter has been changed in the implementation in order to avoid the case $\gamma_1 = 1$, which is a fully unbalance choice for the learning rate parameter.

In Theorem 2.1.1 we have that with appropriate scaling of c_t, i.e. the smoothing parameter for the zeroth order gradient estimator, the iteration dependence of the primal gap matches that of the classical Frank-Wolfe scheme. In particular, for a primal gap of ϵ, the number of iterations needed for the zeroth order scheme in Algorithm 1 is $O\big(\frac{1}{\epsilon}\big)$, while the number of calls to the linear minimization oracle and zeroth order oracle are given by $O\big(\frac{1}{\epsilon}\big)$ and $O\big(\frac{d}{\epsilon}\big)$ respectively.

Consequently, the dimension independence of the primal comes at the cost of querying the zeroth order oracle d times at each iteration. In the sequel, we will focus on the random directions gradient estimator in (5) for the stochastic zeroth order Frank-Wolfe algorithm.

2.2 Stochastic Zeroth Order Frank-Wolfe

When dealing with stochastic approximation of Frank-Wolfe method, a naive replacement of $\nabla f(\boldsymbol{x}_t)$ by its stochastic counterpart $\nabla F(\boldsymbol{x}_t; \boldsymbol{y}_t)$ would make the algorithm divergent due to non-vanishing variance of gradient approximations. In addition, the naive replacement would also lead

Algorithm 2 Stochastic Zeroth Order Frank Wolfe algorithm.

Require: Input, Loss Function $F(x)$, Convex Set $\mathcal{C}$, number of directions m, sequences γ_t, c_t, ρ_t
Output: $\boldsymbol{x}_T$ or $\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t$
Initialize $\boldsymbol{x}_0 \in \mathcal{C}$, $\boldsymbol{d}_0 = 0$.
for $t = 1, 2, \ldots, T$ **do**
    Compute $\boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y})$ with KWSA as in (4) or
        $\boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y}, \boldsymbol{z}_t)$ with RDSA as in (5) or
        $\boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y}, \boldsymbol{z}_t)$ with I-RDSA as in (6)
    Set $\boldsymbol{d}_t = (1-\rho_t)\boldsymbol{d}_{t-1} + \rho_t \boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y}_t)$
    Compute $\boldsymbol{v}_t = \arg\min_{s\in\mathcal{C}}(\boldsymbol{s}\cdot\boldsymbol{d}_t)$
    Set $\boldsymbol{x}_{t+1} = (1-\gamma_t)\boldsymbol{x}_t + \gamma_t \boldsymbol{v}_t$.

to the linear minimization constraint to hold only in expectation and, thereby, likely making the algorithm biased. To solve these problems, the authors (Sahu et al. (2018)) adopt the following method to reduce the algorithm variance:

$$\boldsymbol{d}_t = (1-\rho_t)\boldsymbol{d}_{t-1} + \rho_t \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{y}_t), \tag{15}$$

where $\boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ is a gradient approximation (chosen among the ones described in Section 1.3), $\boldsymbol{d}_0 = \boldsymbol{0}$ and ρ_t is a time-decaying sequence. Furthermore, with this setting, $\mathbb{E}\Big[\, \|\boldsymbol{d}_t - \nabla f(\boldsymbol{x}_t)\|^2 \,\Big]$ goes to zero asymptotically. Then, the linear minimization problem of Frank-Wolfe method and its subsequent steps become

$$\boldsymbol{d}_t = (1-\rho_t)\boldsymbol{d}_{t-1} + \rho_t \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{y}_t), \tag{16}$$

$$\boldsymbol{v}_t = \arg\min_{\boldsymbol{v}\in\mathcal{C}}(\boldsymbol{d}_t \cdot \boldsymbol{v}), \tag{17}$$

$$\boldsymbol{x}_{t+1} = (1-\gamma_{t+1})\boldsymbol{x}_t + \gamma_{t+1}\boldsymbol{v}_t. \tag{18}$$

The authors introduce the following statistics concerning the RDSA and I-RDSA gradient approximations in order to quantify the benefits of such a scheme. For RDSA we have (Duchi et al. (2015))

$$\mathbb{E}_{\boldsymbol{z}_t\sim\mu, \boldsymbol{y}_t\sim\mathcal{P}}[\boldsymbol{g}(\boldsymbol{x}; \boldsymbol{y}_t, \boldsymbol{z}_t)] = \nabla f(\boldsymbol{x}) + c_t L \boldsymbol{v}(\boldsymbol{x}, c_t), \tag{19}$$

$$\mathbb{E}_{\boldsymbol{z}_t\sim\mu, \boldsymbol{y}_t\sim\mathcal{P}}\big[\, \|\boldsymbol{g}(\boldsymbol{x}; \boldsymbol{y}_t, \boldsymbol{z}_t)\|^2 \,\big] \leq 2s(d)\mathbb{E}\big[\, \|\nabla F(\boldsymbol{x}; \boldsymbol{y}_t)\|^2 \,\big] + \frac{c_t^2}{2} L^2 M(\mu), \tag{20}$$

and using (19) and (20) for I-RDSA similar statistics can be obtained:

$$\mathbb{E}_{\boldsymbol{z}_t\sim\mu, \boldsymbol{y}_t\sim\mathcal{P}}[\boldsymbol{g}_m(\boldsymbol{x}; \boldsymbol{y}_t, \boldsymbol{z}_t)] = \nabla f(\boldsymbol{x}) + \frac{c_t}{m} L \boldsymbol{v}(\boldsymbol{x}, c_t), \tag{21}$$

$$\mathbb{E}_{\boldsymbol{z}_t\sim\mu, \boldsymbol{y}_t\sim\mathcal{P}}\big[\, \|\boldsymbol{g}_m(\boldsymbol{x}; \boldsymbol{y}_t, \boldsymbol{z}_t)\|^2 \,\big] \leq \Big(\frac{1+m}{2m}\Big) c_t^2 L^2 M(\mu) + 2\Big(1 + \frac{s(d)}{m}\Big)\mathbb{E}\big[\, \|\nabla F(\boldsymbol{x}; \boldsymbol{y}_t)\|^2 \,\big], \tag{22}$$

where $\|\boldsymbol{v}(\boldsymbol{x}, c_t)\| \leq \frac{1}{2}\mathbb{E}\Big[\, \|\boldsymbol{z}\|^3 \,\Big]$.

In Algorithm 2 the stochastic zeroth order Frank-Wolfe by (Sahu et al. (2018)) is presented, with the three zeroth order oracles previously reported for the estimation of $\boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y})$ or $\boldsymbol{g}(\boldsymbol{x}_t; \boldsymbol{y}, \boldsymbol{z}_t)$. The following Lemma reports the study of the error for $\boldsymbol{d}_t$ as defined in (15).

Lemma 2.2.1. *Let Assumptions 1-5 hold. Given the recursion in (15), we have that* $\|\nabla f(\boldsymbol{x}_t - \boldsymbol{d}_t)\|^2$ *satisfies*

1. *for the RDSA*

$$\mathbb{E}\Big[\ \|\nabla f(\boldsymbol{x}_t) - \boldsymbol{d}_t\|^2\ \Big] \leq 2\rho_t^2\sigma^2 + 4\rho_t^2 L_1^2 + 8\rho_t^2 s(d) L_1^2 + 2\rho_t^2 c_t^2 L^2 M(\mu)$$
$$+ \frac{2L^2R^2\gamma_t^2}{\rho_t} + \frac{\rho_t}{2}c_t^2 L^2 M(\mu) + \Big(1 - \frac{\rho_t}{2}\Big)\mathbb{E}\Big[\ \|\nabla f(\boldsymbol{x}_{t-1}) - \boldsymbol{d}_{t-1}\|^2\ \Big],$$

2. *for the I-RDSA*

$$\mathbb{E}\big[\ \|\nabla f(\boldsymbol{x}_t) - \boldsymbol{d}_t\|^2\ \big] \leq 2\rho_t^2(\sigma^2 + 2L_1^2) + \frac{\rho_t}{2m^2}c_t^2 L^2 M(\mu) + 8\rho_t^2\Big(1 + \frac{s(d)}{m}\Big)L_1^2$$
$$+ \Big(\frac{1+m}{2m}\Big)\rho_t^2 c_t^2 L^2 M(\mu) + \frac{2L^2R^2\gamma_t^2}{\rho_t} + \Big(1 - \frac{\rho_t}{2}\Big)\mathbb{E}\big[\ \|\nabla f(\boldsymbol{x}_{t-1}) - \boldsymbol{d}_{t-1}\|^2\ \big],$$

3. *for KWSA*

$$\mathbb{E}\big[\ \|\nabla f(\boldsymbol{x}_t) - \boldsymbol{d}_t\|^2\ \big] \leq 2\rho_t^2\sigma^2 + 2\rho_t c_t^2 dL^2 + \frac{2L^2R^2\gamma_t^2}{\rho_t} + \Big(1 - \frac{\rho_t}{2}\Big)\mathbb{E}\big[\ \|\nabla f(\boldsymbol{x}_{t-1}) - \boldsymbol{d}_{t-1}\|^2\ \big].$$

Now we can analyse the convergence of the Algorithm 2.

Lemma 2.2.2. *If assumptions 1-5 hold for Algorithm 2, then the primal gap $F(\boldsymbol{x}_{t+1} - F(\boldsymbol{x}^*))$ satisfies*

$$F(\boldsymbol{x}_{t+1} - F(\boldsymbol{x}^*)) \leq (1 - \gamma_{t+1})(F(\boldsymbol{x}_t - F(\boldsymbol{x}^*)) + \gamma_{t+1}R\,\|\nabla F(\boldsymbol{x}_t) - \boldsymbol{d}_t\| + \frac{LR^2\gamma_{t+1}^2}{2}.$$

In the following we report the results involving different gradient approximation schemes for the primal gap, i.e. $\mathbb{E}[f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)]$

Theorem 2.2.3. *Let Assumptions 1-5 hold. If the sequence γ_t is set to $\gamma_t = \frac{2}{t+8}$*

1. *then, we have the following primal sub-optimality gap for the Algorithm 2, with the RDSA gradient approximation scheme*

$$\mathbb{E}[f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)] = O\bigg(\frac{d^{1/3}}{(t+9)^{1/3}}\bigg), \tag{23}$$

2. *in case of I-RDSA the gradient approximation scheme, the primal sub-optimality gap is given by,*

$$\mathbb{E}[f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)] = O\bigg(\frac{(d/m)^{1/3}}{(t+9)^{1/3}}\bigg). \tag{24}$$

3. *Finally, for the KWSA gradient approximation scheme, the primal sub-optimality gap is given by,*

$$\mathbb{E}[f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)] = O\bigg(\frac{1}{(t+9)^{1/3}}\bigg). \tag{25}$$

Algorithm 3 Inexact Conditional Gradient (ICG) Method

Require: $\boldsymbol{v}, \boldsymbol{g}, \gamma, \mu$
Output: $\bar{\boldsymbol{y}}_t$
Set $\bar{\boldsymbol{y}}_t = \boldsymbol{v}$, $t = 1$ and $\kappa = 0$
while $\kappa = 0$ **do**
    $\boldsymbol{y}_t = \arg\min_{\boldsymbol{u}\in\mathcal{C}} \{h_\gamma(\boldsymbol{u}) \coloneqq [(\boldsymbol{g} + \gamma(\bar{\boldsymbol{y}}_{t-1} - \boldsymbol{v})) \cdot (\boldsymbol{u} - \boldsymbol{y}_{t-1})]\}$
    if $h_\gamma(\boldsymbol{y}_t) \geq -\mu$ **then**
        Set $\kappa = 1$ and $\boldsymbol{v}_t = \bar{\boldsymbol{y}}_t$
    else
        Set $\bar{\boldsymbol{y}}_t = \frac{t-1}{t+1}\bar{\boldsymbol{y}}_{t-1} + \frac{2}{t+1}\boldsymbol{y}_t$ and $t = t + 1$

Theorem 2.2.3 quantifies the dimension dependence of primal gap to be $d^{1/3}$. At the same time the dependence on iterations, i.e. $O(1/T^3)$ 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 so as to obtain a primal gap of ϵ, i.e. $\mathbb{E}[f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)] \leq \epsilon$ is given by $O\big(\frac{d}{\epsilon^3}\big)$, where the dimension dependence is consistent with zeroth order schemes and cannot be improved (Duchi et al. (2015)).

Finally, we report the values of sequences c_t and ρ_t used for the following experiments, which are the same used by authors for their paper (Sahu et al. (2018)):

$$(\rho_t, c_t)_{\text{RDSA}} = \left(\frac{4}{d^{1/3}(t+8)^{2/3}}, \frac{2}{d^{3/2}(t+8)^{1/3}}\right), \tag{26}$$

$$(\rho_t, c_t)_{\text{I-RDSA}} = \left(\frac{4}{(1+\frac{d}{m})^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}}\right), \tag{27}$$

$$(\rho_t, c_t)_{\text{KWSA}} = \left(\frac{4}{(t+8)^{2/3}}, \frac{2}{d^{1/2}(t+8)^{1/3}}\right). \tag{28}$$

2.3 Zeroth order Stochastic Accelerated Gradient Method with Inexact Updates

In this section we present the algorithm proposed by (Balasubramanian and Ghadimi (2018)). We first introduce a subroutine that allows us to compute an inexact conditional gradient (ICG) in Algorithm 3, then the main algorithm zeroth-order stochastic accelerated gradient method with inexact updates is given by Algorithm 4. The ICG basically substitute the arg min problem solve in the "exact" versions of Frank-Wolfe. This allows, with parameters suitably chosen, to reduce the number of operations done in the whole algorithm.

Notice that Algorithm 3 is equivalent to solve the problem

$$\mathbb{P}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{g}, \gamma) = \arg\min_{\boldsymbol{u}\in\mathcal{C}} \left\{(\boldsymbol{g} \cdot \boldsymbol{u}) + \frac{\gamma}{2}||\boldsymbol{u} - \boldsymbol{x}||^2\right\}. \tag{29}$$

Algorithm 4 Zeroth-order Stochastic Accelerated Gradient Method with Inexact Updates

Require: Input, Loss Function $F(x)$, smoothing parameter $c > 0$, sequences $\alpha_t, m_t, \gamma_t, \mu_t$ and iteration limit T
Output: $\boldsymbol{x}_T$ or $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t$
Initialize $\boldsymbol{x}_0 \in \mathcal{C}$
for $t = 1, ..., T$ **do**
    Set
$$\boldsymbol{d}_t = (1 - \alpha_t)\boldsymbol{x}_{t-1} + \alpha_t \boldsymbol{v}_{t-1}$$
    Compute $\boldsymbol{g}(\boldsymbol{d}_t; \boldsymbol{y}, \boldsymbol{z}_t)$ with I-RDSA as in (6)
    Compute $\boldsymbol{v}_t$ with ICG as in Algorithm 3 using $\boldsymbol{v}_{t-1}$, $\boldsymbol{g}(\boldsymbol{d}_t; \boldsymbol{y}, \boldsymbol{z}_t)$, γ_t, μ_t
    Set
$$\boldsymbol{x}_t = (1 - \alpha_t)\boldsymbol{x}_{t-1} + \alpha_t \boldsymbol{v}_t$$

Theorem 2.3.1. *Let $\{\boldsymbol{x}_t\}_{t\geq 1}$ be generated by Algorithm 4, the function f be convex and*

$$\alpha_t = \frac{2}{t+1},\ \gamma_t = \frac{4L}{t},\ \mu_t = \frac{LD_X^0}{tT},\ c = \frac{1}{\sqrt{2T}} \max\left\{\frac{1}{d+3}, \sqrt{\frac{D_X^0}{d(T+1)}}\right\},$$
$$m_t = \frac{t(t+1)}{D_X^0} \max\{(d+5)B_{L\sigma}T, d+3\},\ \forall t \geq 1,$$

and for some constants $D_X^0 \geq \|\boldsymbol{x_0} - \boldsymbol{x}^*\|^2$ and $B_{L\sigma} \geq \max\{\sqrt{B^2 + \sigma^2}/L, 1\}$. Then under Assumptions 1-2-3, we have

$$\mathbb{E}[f(\boldsymbol{x}_T) - f(\boldsymbol{x}^*)] \leq \frac{12LD_X^0}{T(T+1)}. \tag{30}$$

Hence, the total number of calls to the stochastic oracle and linear sub-problems solved to find an ϵ-stationary point of problem (1) are respectively bounded by $O\left(\frac{d}{\epsilon^3}\right)$ and $O\left(\frac{1}{\epsilon}\right)$.

Notice that the α_t parameter has been changed in the implementation in order to avoid the case $\alpha_1 = 1$, which is a fully unbalance choice for the learning rate parameter. In addition, it is important to underline that here the parameter m_t depends on T : maximum number of iterations and d : dimension of the problem. In the previous algorithms this parameter was fixed to $m_t = m$ and, obviously, independent from that quantities.

3 Experiments

In order to show that the presented methods are accurate even in the stochastic case and scale to relatively high dimensions, we study the performance of the algorithms previously described when solving a stochastic Lasso regression and an high dimensional Cox regression problem.

In this framework we will refer to the variable of the problem as $\boldsymbol{w}$, while $\boldsymbol{x}$ will represent the available data.

We look at the optimality gap $\|f(\boldsymbol{w}_{\text{optimizer}}) - f(\boldsymbol{w}^*)\|$ as evaluation metric, where $\boldsymbol{w}_{\text{optimizer}}$ denotes the solution obtained from the employed optimizer and $\boldsymbol{w}^*$ corresponds to true solution.

Finally, in Tab. 1 the values of the parameters for each analysed algorithm of the following sections are reported.

Table 1: Values of the parameters for each algorithm.

	Covtype dataset	**Artificial dataset**	**Kidney dataset**
DZFW	L: biggest eigenvalue of the Hessian matrix, $\boldsymbol{w}_0$: origin of the axes, $\varepsilon = 1e-8$.	L: biggest eigenvalue of the Hessian matrix, $\boldsymbol{w}_0$: sparse vector, $\varepsilon = 1e-8$.	$L = 0.1$, $\boldsymbol{w}_0$: random vector, $\varepsilon = 1e-8$.
SZFW	$\boldsymbol{w}_0$: origin of the axes, $\varepsilon = 1e-8$, $m = 1$ with RDSA, $m = 6$ with I-RDSA.	$\boldsymbol{w}_0$: sparse vector, $\varepsilon = 1e-8$, $m = 1$ with RDSA, $m = 100$ with I-RDSA.	$\boldsymbol{w}_0$: random vector, $\varepsilon = 1e-8$, $m = 1$ with RDSA, $m = 938$ with I-RDSA.
IZFW	$L = 0.01$, $D_X^0 = 1e4$, $B = 1$, $\boldsymbol{w}_0$: origin of the axes, $\varepsilon = 1e-6$.	$L = 0.1$, $D_X^0 = 1e4$, $B = 1$, $\boldsymbol{w}_0$: random vector, $\varepsilon = 1e-8$.	$L = 0.1$, $D_X^0 = 1e4$, $B = 1$, $\boldsymbol{w}_0$: random vector, $\varepsilon = 1e-6$.

3.1 Stochastic Lasso Regression

To study the performance of various stochastic optimization methods, we solve a low dimensional Lasso regression on the *covtype dataset* ($n = 581012$, $d = 54$) from libsvm website available here, so as proposed in the background paper. We use the variant with feature values in $[0, 1]$ and solve the following problem

$$\min_{\|\boldsymbol{w}\|_1 \leq 1} \frac{1}{2n} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2, \tag{31}$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ represents the feature vectors and $\boldsymbol{y} \in \mathbb{R}^n$ are the corresponding targets.

However, since the Lasso framework is adopted when $n < d$, which is not the case examined by the authors, we have also built an artificial dataset respecting this condition.

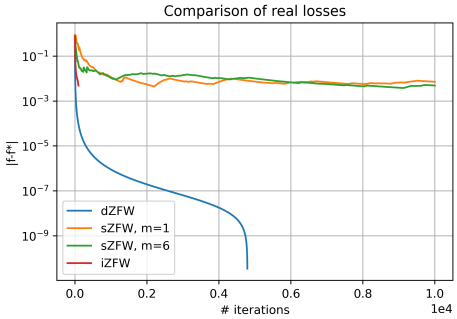
In order to build the *artificial dataset*, we define the feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, ($n = 100$, $d = 1000$) to be a Gaussian matrix and a sparse vector $\boldsymbol{w}^* \in \mathbb{R}^d$ with only 50 non-zero elements. We compute $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}^* + \boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon}$ small random noise. We will use the proposed algorithms to retrieve a solution $\boldsymbol{w}$ by solving (31). Notice that for this case, when evaluating the optimality gap $\|f(\boldsymbol{w}_{\text{optimizer}}) - f(\boldsymbol{w}^*)\|$ we know the optimal solution, while for the previous dataset we will consider $f(\boldsymbol{w}^*)$ to be the minimum output returned by the considered algorithms.

3.1.1 Covtype dataset analysis

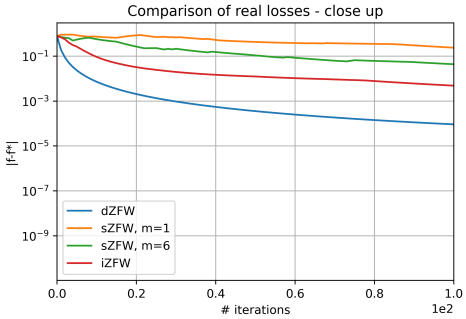
Herein the results for the Covtype dataset are reported, they have been obtained by using the Google Colab platform.

The algorithms examined are: DZFW, SZFW with RDSA, SZFW with I-RDSA ($m = 6$, i.e approximately 10% of the problem dimension d), IZFW with $m_t < d$.

The parameters have been chosen according to Theorems 2.1.1, 2.2.3, 2.3.1. When the Lipschitz continuity constant is required, we approximate it with the biggest eigenvalue of the Hessian matrix

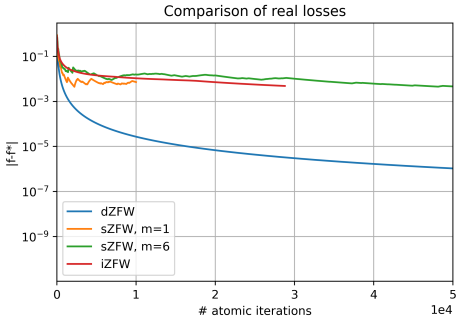


(a) All iterations shown.

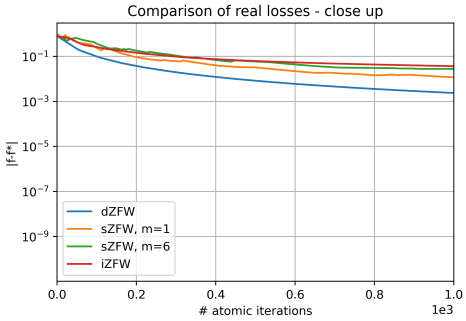


(b) Close up on the set of iterations in common.

Figure 1: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Covtype dataset.



(a) All atomic iterations shown.



(b) Close up on the set of atomic iterations in common.

Figure 2: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Covtype dataset.

calculated for any $\boldsymbol{w}$ or we suitably choose it according to the specific algorithm requirements. A resume is shown in Tab. 1.

The comparison between the real losses computed at each iteration is shown in Figure 1. Herein, we do not take into account all the inner iterations done by the algorithms, i.e. we have considered the gradient approximation as an oracle. It is important to notice that the SZFW with RDSA has a performance really close to the one of the SZFW with I-RDSA and $m = 6$, probably because this dataset has a small value of d.

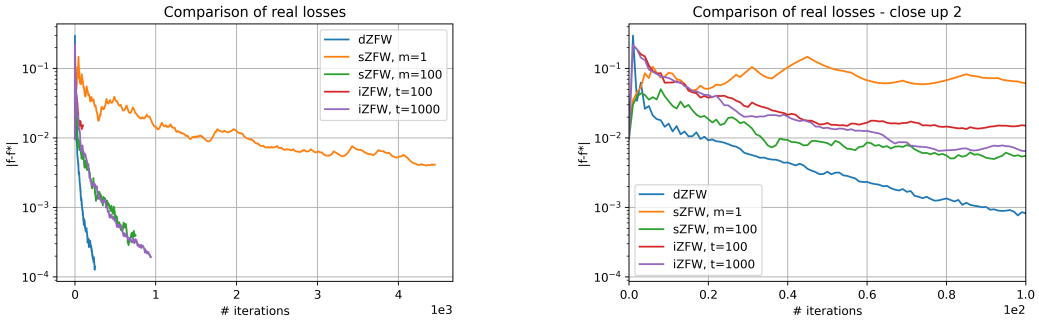
As regards the IZFW, it reaches results similar to the ones obtained by SZFW (with both RDSA and I-RDSA) in a lower number of iterations. However, all the stochastic oracles have a poor performance if compared to the DZFW.

In Figure 2 we take into account all the performed iterations (*atomic iterations*), hence also the ones done inside the gradient oracle (dependent on m or d) and all the iterations of the `while` loop in the Inexact algorithm. With this representation, we point out that in the first atomic iterations the algorithms have similar performances, but again the DFZW is faster in reaching a better result. IZFW and SZFW have really similar results.

Nevertheless, we conclude that the results for this experiment are quite strange: the Lasso

problem is applied to a dataset with $n > d$, so that we cannot observe the typical sparsification phenomena; the results reached with our experiments (SZFW with I-RDSA) are similar in shape but not in accuracy, even if the number of atomic iterations is comparable with the one of the paper (Sahu et al. (2018)).

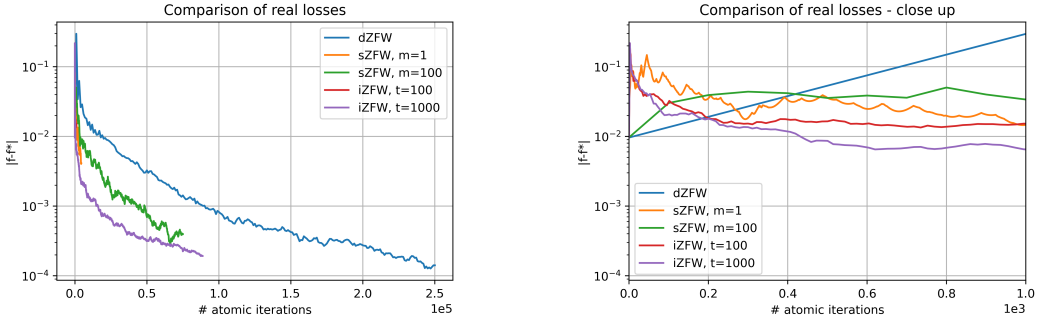
The difference in the quality of the convergence could be due to the fact that, for the Covtype dataset, we do not have any knowledge about the real solution. Hence, it has been impossible for us to run the experiments starting from an initial point close to the optimal solution.



(a) All iterations shown.

(b) Close up on the set of iterations in common.

Figure 3: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Artificial dataset.



(a) Almost all atomic iterations shown.

(b) Close up on the set of atomic iterations in common.

Figure 4: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Artificial dataset.

3.1.2 Artificial Dataset

In this section the results obtained for the Artificial dataset are reported, collected by using our personal computers.

The algorithms examined are: DZFW, SZFW with RDSA, SZFW with I-RDSA ($m = 100$, i.e. the 10% of the problem dimension d), IZFW with $m_t < d$ (case with 100 iterations) and IZFW with $m_t > d$ (case with 1000 iterations).

The parameters have been chosen according to Theorems 2.1.1, 2.2.3, 2.3.1. When the Lipschitz continuity constant is required, we approximate it with the largest eigenvalue of the Hessian matrix

calculated for any $\boldsymbol{w}$ or we suitably choose it according to the specific algorithm requirements. A resume is shown in Tab. 1.

In general, it makes more sense to define the m_t parameter such that $m_t < d$ for most of the values of t. This because when happens that $m_t > d$ for an high number of iterations, there would be a significant increase in the computational time of IZFW. In this work we adopt both $m_t < d$ and $m_t > d$ for approximately the 50% of the iterations in order to study both the solutions.

In Figure 3 the results reached by the algorithms for the Artificial dataset are reported.

It can be observed that DZFW has again the best performance in terms of accuracy and number of iterations used. Some of the stochastic versions, such as IZFW with $m_t > d$ and SZFW with I-RDSA reach similar values for the loss evaluation, but with an higher number of steps. Moreover, since d now assumes a larger value with respect to the previous experiment, there is a drop in the performances of SZFW with RDSA.

In Figure 3 we represent the results taking into account the atomic iterations.

The DZFW clearly needs a number of atomic steps not feasible if compared to the accuracy it reaches. The stochastic solutions seems to be more interesting implementations, in particular when an acceptable solution must be reached in a reasonable amount of time. In this case, the IZFW with $m_t < d$ and SZFW with RDSA is preferable.

Finally, the IZFW with $m_t > d$ has a good performance in the considered window, but we cannot modify the maximum number of iterations T without causing a change in the performances, since T influences some parameters (see Theorem 2.3.1).

3.2 High Dimensional Cox Regression

To demonstrate the efficacy of zeroth order Frank-Wolfe optimization in a moderately high dimensional case, we investigate its performance when applied to gene expression data. In particular, we perform patient survival analysis by solving Cox regression (also known as proportional hazards regression) to relate different gene expression profiles with survival times.

The Kidney renal clear cell carcinoma dataset (available here) is used. It contains gene expression data for 606 patients (534 with tumor and 72 without tumor) along with survival time information (available only for part of them).

The dataset is pre-processed by eliminating the rarely expressed genes, i.e. we only keep genes expressed in 50% of the patients with a score of at least 275.735. This leads to a feature vector $\boldsymbol{x}_i$ of size 9375 for each patient i of the 533 ones with survival time information available. In addition, for each patient i, we also consider the censoring indicator variable y_i which takes value 0 if the patient is alive or 1 if dead with t_i denoting the time of death.

In this setup, we can obtain a sparse solution to Cox regression by solving the problem

$$\min_{\|\boldsymbol{w}\|_1 \leq 10} \frac{1}{n} \sum_{i=1}^{n} \left\{ - \boldsymbol{x}_i^\top \boldsymbol{w} + \log \left(\sum_{j \in \mathcal{R}_i} \exp(\boldsymbol{x}_i^\top \boldsymbol{w}) \right) \right\}, \tag{32}$$

where $\mathcal{R}_i = \{j \mid t_j \geq t_i\}$. This problem represents an high-dimensional setting, since $d = 9375$.

The results presented in Figures 5 and 6 have been obtained by using the cluster of computers provided by the Computer Science Department of UPC.

The algorithms examined are: DZFW, SZFW with RDSA, SZFW with I-RDSA ($m = 938$, i.e. 10% of the problem dimension d), IZFW with $m_t < d$.

The parameters have been chosen according to Theorems 2.1.1, 2.2.3, 2.3.1. When the Lipschitz continuity constant is required, we suitably choose it depending of the specific algorithm requirements. A resume is shown in Tab. 1.

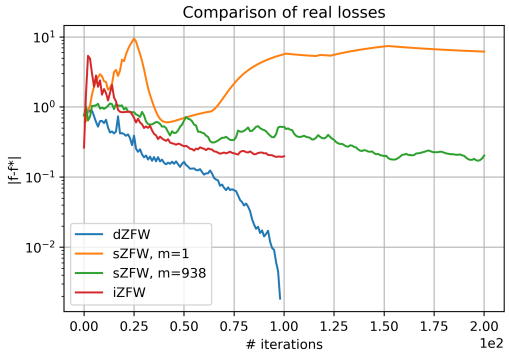
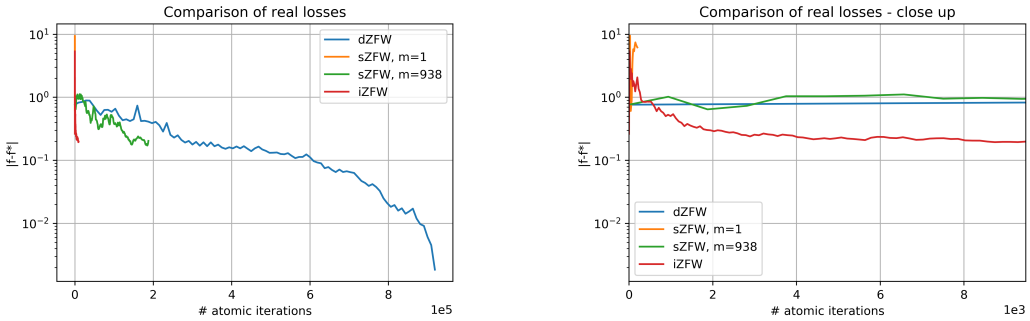


Figure 5: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Kidney dataset evaluated in terms of number of iterations.



(a) All atomic iterations shown.

(b) Close up on the set of atomic iterations in common.

Figure 6: Comparisons of the performances (real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$) on the Kidney dataset.

Looking at Figure 5, we clearly see that the SZFW with RDSA is poorly performing on the high dimensional dataset. The I-RDSA variant has better accuracy values, but it is outperformed by the IZFW algorithm. However, the best results is again achieved when using the deterministic algorithm.

When considering the atomic iterations (Fig. 6), the DZFW has a too high number of iterations needed to reach its optimal solution. The SZFW with I-RDSA behaves similarly in the first steps, finishing in a lower number of iterations. The IZFW stops in the lowest number of steps (1000), reaching the best result among the stochastic versions of ZFW.

4 Conclusions

The experiments done in this report allow us to give a description of how these algorithms are expected to behave on datasets having high or low dimensions.

In both cases the Deterministic variant of ZFW has the best result, but as long as d increases it becomes unfeasible to use, due to the huge number of atomic iterations needed. When evaluating

the most simple algorithm solving this problem, i.e. the SZFW with RDSA, we have seen that it is efficient but imprecise: the experiments suggests that it should be adopted used only if $m = 1$ is near to the 10% of d' s value and when a rougher and quicker solution is preferred to an accurate one with a high time cost.

In all the other cases, in particular in the high dimensional setting, it seems that IZFW can be a good trade-off between accuracy and computational time.

When $m_t < d$ for enough iterations t, the results achieved with IZFW are similar to the SZFW with I-RDSA, but we highlight that they are reached in a lower or similar number of atomic steps.

References

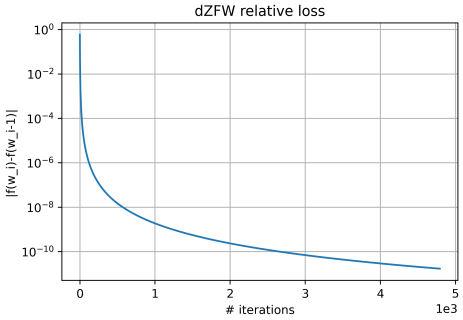
Balasubramanian, K. and Ghadimi, S. (2018). Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates. In *Advances in Neural Information Processing Systems 31*, pages 3455–3464. Curran Associates, Inc.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono., A. (2015). Optimal rates for zero-order convex optimization: the power of two function evaluations. IEEE Transactions on Information Theory.

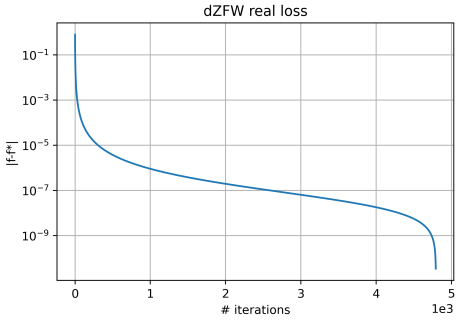
Sahu, A. K., Zaheer, M., and Kar, S. (2018). Towards gradient free and projection free stochastic optimization.

A Appendix: Additional Figures

A.1 Covtype dataset

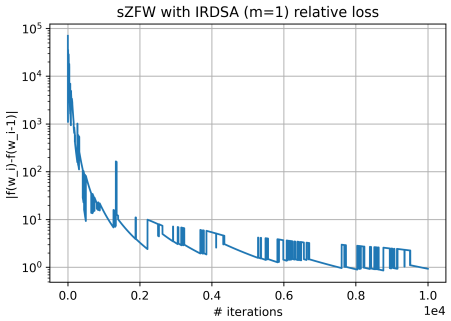


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

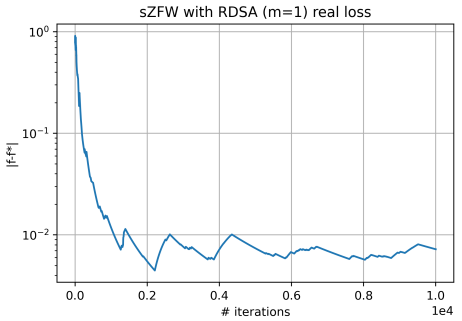


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 7: Performances for the DZFW applied to the Covtype dataset.

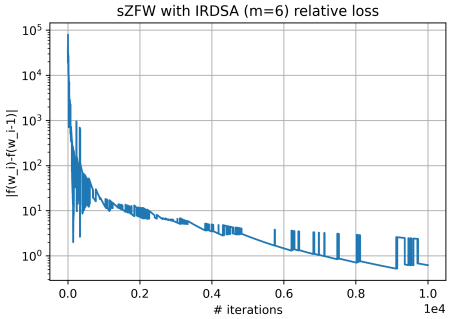


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

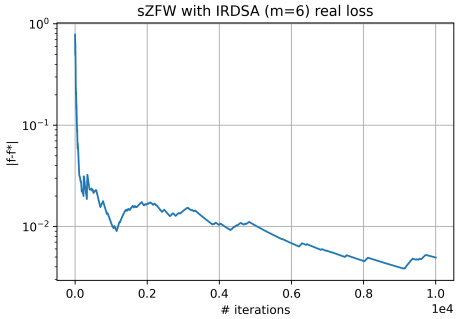


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 8: Performances for the SZFW with RDSA applied to the Covtype dataset.

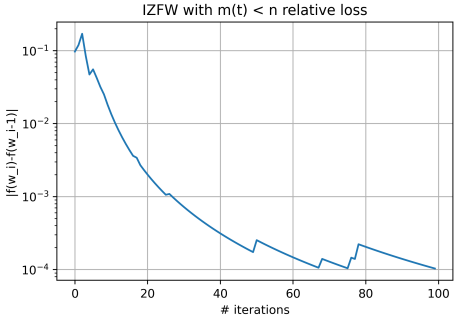


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

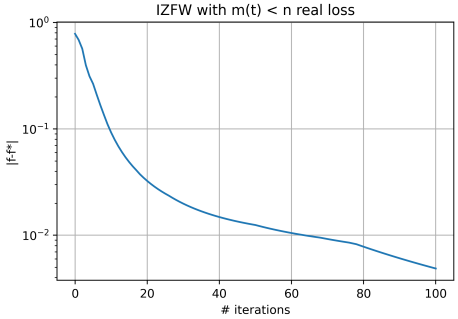


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 9: Performances for the SZFW with I-RDSA applied to the Covtype dataset.



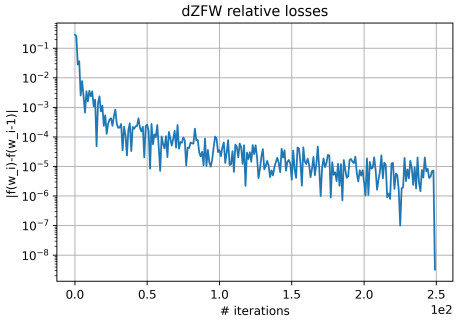
(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.



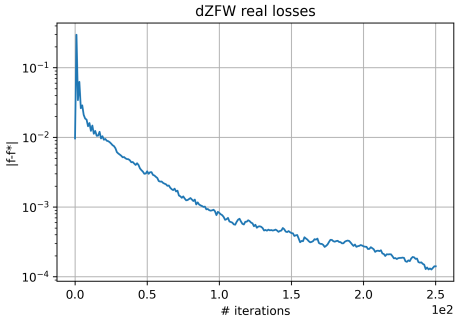
(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 10: Performances for the IZFW applied to the Covtype dataset.

A.2 Artificial dataset

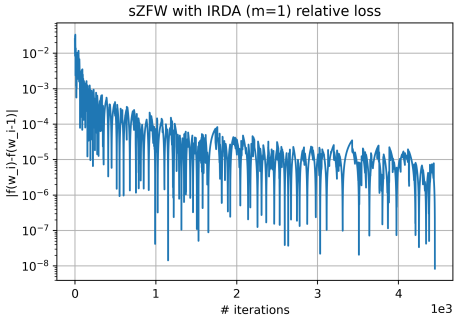


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

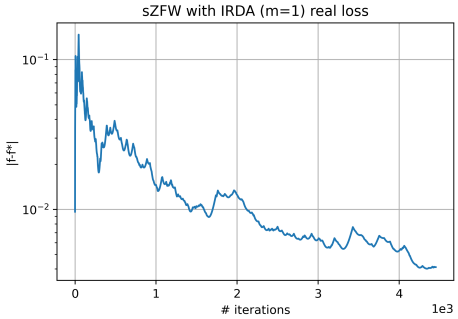


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 11: Performances for the DZFW applied to the Artificial dataset.



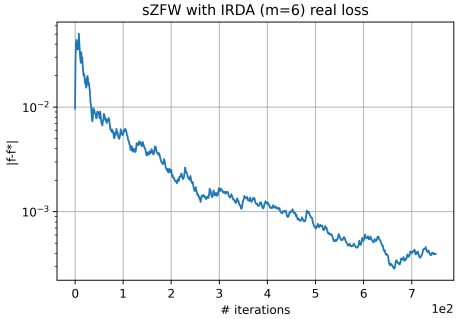
(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.



(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

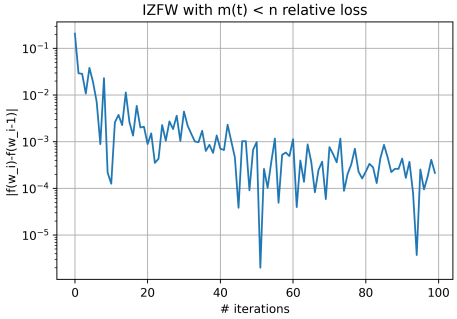
Figure 12: Performances for the SZFW with RDSA applied to the Artificial dataset.

(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

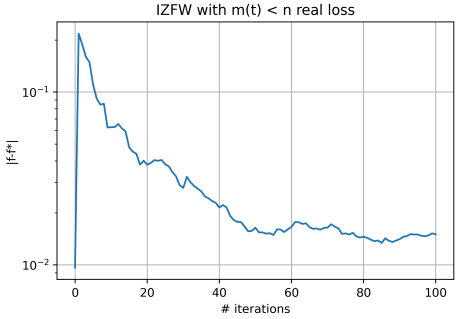


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 13: Performances for the SZFW with I-RDSA applied to the Artificial dataset.



(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

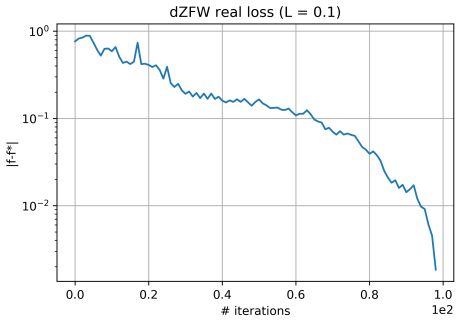


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 14: Performances for the IZFW applied to the Artificial dataset.

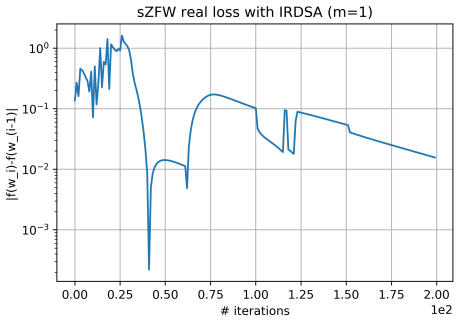
A.3 Kidney (Cox) dataset

(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

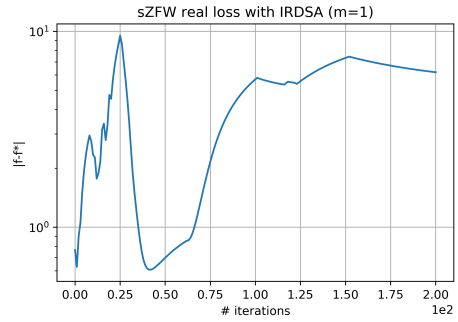


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 15: Performances for the DZFW applied to the Kidney dataset.

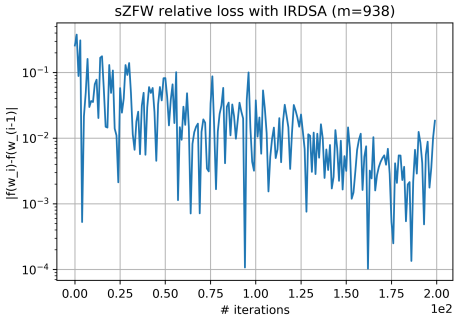


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

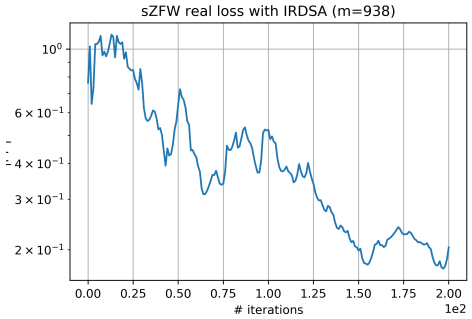


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 16: Performances for the SZFW with RDSA applied to the Kidney dataset.

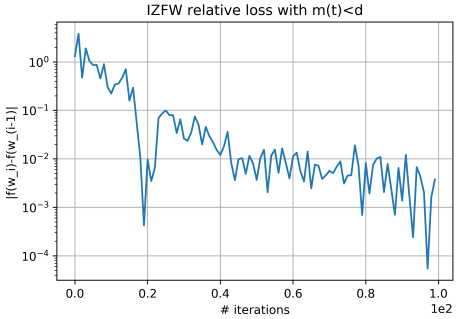


(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.

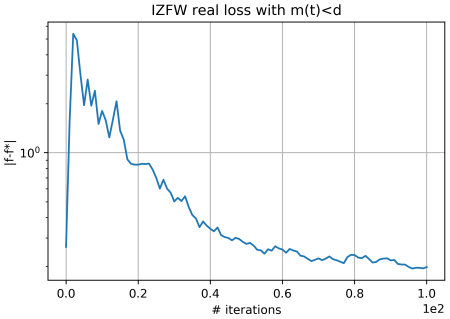


(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 17: Performances for the SZFW with I-RDSA applied to the Kidney dataset.



(a) Relative losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}_{t-1})\|$.



(b) Real losses, $\|f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)\|$.

Figure 18: Performances for the IZFW applied to the Kidney dataset.