# First order projection-free optimization in the stochastic conditional gradient methods framework

Pietro Maria Sanguin
pietromaria.sanguin@studenti.unipd.it

Emma Roveroni
emma.roveroni@studenti.unipd.it

Marco Tesser
marco.tesser.1@studenti.unipd.it

Angela Bissacco
angela.bissacco.1@studenti.unipd.it

## Abstract

*In this paper, we present and analyze different projection-free stochastic optimization methods. The algorithms considered include:*

1. ***Stochastic Frank-Wolfe (SFW)**: the simplest variant of the classic Frank-Wolfe method for optimizing constrained problems, which substitutes the full gradient with a stochastic estimate;*
2. ***Stochastic Conditional Gradient (SCG)**: a biased SFW variant which exploits an average gradient computation to reduce asymptotically the gradient estimate variance;*
3. ***One-sample Stochastic Frank-Wolfe (1-SFW)**: an improved SCG version, which also manages get an unbiased gradient estimate;*
4. ***Stochastic Variance-Reduced Franck-Wolfe (SVRF)**: another unbiased variance reduced FW variant which exploits also some full gradient computations to improve the method.*

*In the following table are presented the results of the algorithm in terms of iterations to get $1-\epsilon$ accuracy.*

| Algo. | #Exact G. | #Stoch. G. | #Lin. Opt. |
|---|---|---|---|
| SFW | 0 | $\mathcal{O}(1/\epsilon^3)$ | $\mathcal{O}(1/\epsilon)$ |
| SCG | 0 | $\mathcal{O}(1/\epsilon^3)$ | $\mathcal{O}(1/\epsilon^3)$ |
| 1-SFW | 0 | $\mathcal{O}(1/\epsilon^2)$ | $\mathcal{O}(1/\epsilon^2)$ |
| SVRF | $\mathcal{O}(\log_2(\frac{LD^2}{\epsilon}))$ | $\mathcal{O}(\frac{L^2D^4}{\epsilon^2})$ | $\mathcal{O}(\frac{LD^2}{\epsilon})$ |

Table 1: Summary of the different F.W. variants reporting the number of Exact Gradients, Stochastic Gradients and Linear Optimizations required, respectively.

## 1. Introduction

Due to their structure, convex optimization problems are inherently easy to optimize since they have a precisely identifiable set of global minima. Stochastic optimization methods are very useful when working with large datasets since they take less time than classic deterministic methods to perform a single gradient step. As shown in [4] the goal is to solve the problem:

$$\min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}) := \min_{\mathbf{x}\in\mathcal{C}} \mathbb{E}_{z\sim P}[\tilde{F}(\mathbf{x},\mathbf{z})] \tag{1}$$

In particular, an optimization variable $\mathbf{x} \in \Omega \subset \mathbb{R}^n$ and a random variable $\mathbf{Z} \in \mathcal{Z}$ together determine the choice of a stochastic function $\tilde{F} : \Omega \times \mathcal{Z} \to \mathbb{R}$. Moreover $F$ is convex and subject to the convex constraint $\mathcal{C}$ and $\mathbf{z}$ is the realization of the random variable $\mathbf{Z}$ drawn from a distribution $P$. Note that the main challenge here is to solve problem (1) without computing the objective function $F(\mathbf{x})$ or its gradient $\nabla F(\mathbf{x})$ explicitly, since we assume that either the probability distribution $P$ is unknown or the cost of computing the expectation is prohibitive. The efficiency of a projection-free algorithm is measured by the number of exact gradient evaluations, stochastic gradient evaluations and linear optimizations which are needed to achieve $1-\epsilon$ accuracy, that is to output a point $x_t \in \mathcal{C}$ such that $\mathbb{E}[F(x_t) - F(x^*)] \leq \epsilon$ where $x^* \in \arg\min_{\mathbf{x}\in\Omega} F(x)$ is any optimum.

To study the algorithm's convergence rate we assume the following hypotheses to hold:

1. **Bounded constraint** $\mathcal{C}$: $\forall \mathbf{x}, \mathbf{y} \in \mathcal{C}$

$$\|\mathbf{x} - \mathbf{y}\| \leq D \tag{2}$$

2. **F convex**: we deal with convex functions

$$\nabla F(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \leq F(\mathbf{y}) - F(\mathbf{x}) \tag{3}$$

3. **Lipschitz Continuous Gradient (LCG)**: a LCG function over $\mathcal{C}$ sometimes is also called L-smooth

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \tag{4}$$

It can be proved that an equivalent form of equation (4) is:

$$F(\mathbf{x}) \leq F(\mathbf{y}) + \nabla F(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2 \tag{5}$$

For some proofs it is useful to add also the following hypothesis:

4. **Bounded variance**:

$$\mathbb{E}\left[\|\nabla \tilde{F}(\mathbf{x}, \mathbf{z}) - F(\mathbf{x})\|^2\right] \leq \sigma^2 \tag{6}$$

## 2. The SFW algorithm

In this section we are going to briefly analyze the main characteristics of Stochastic Frank-Wolfe algorithm.
We denote the stochastic gradient of $F$ at some $\mathbf{x}$ as $\nabla F_i(\mathbf{x})$ where $i$ is picked from $\{1, \ldots, n\}$ uniformly at random. This FW variant simply replaces the full gradient $\nabla F(x_{t-1})$ with some stochastic $\nabla F_i(x_{t-1})$ variant or more generally the average of some number of i.i.d. samples of $\nabla F_i(x_{t-1})$ (mini-batch approach), as stated in [3]. Next we will denote by $\tilde{\nabla}_t$ the average of $m_t$ i.i.d. samples of stochastic gradients $\nabla F_i(\mathbf{x}_{t-1})$. To prove the convergence of the algorithm is done also this additional hypothesis:

5. $\mathbf{F_i}$ **is G-Lipschitz**:

$$\|\nabla F_i(\mathbf{x})\| \leq G \qquad \forall \mathbf{x} \in \Omega \tag{7}$$

**Algorithm 1** Stochastic Frank-Wolfe (SFW)

**Require:** Stepsize $\gamma_t = \frac{2}{t+1} \in (0, 1]$, $m_t = \left(\frac{G(t+1)}{LD}\right)^2$. Choose $\mathbf{x}_1 \in \mathcal{C}$
**for** $t = 1, 2, \ldots T$ **do**
1: Sample $i \in \{1, \ldots, n\}$ uniformly at random
2: Find descent direction with stochastic gradient $\mathbf{v}_t = \underset{\mathbf{v}\in\Omega}{\operatorname{argmin}}\{\tilde{\nabla}_t^T \mathbf{v}\}$
3: Compute the updated variable $\mathbf{x}_{t+1} = (1 - \gamma_{t+1})\,\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t$
**end for**

### 2.1. Convergence analysis

**Theorem 2.1 (SFW Convergence Rate)** *If conditions 1-3 hold and each $F_i$ is G-Lipschitz, we set $\gamma_t = \frac{2}{k+1}$ and $m_t = \left(\frac{G(t+1)}{LD}\right)^2$, then we have:*

$$\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \frac{4LD^2}{(t+2)} \tag{8}$$

Therefore the SFW needs $\mathcal{O}(\frac{LD^2}{\epsilon})$ iterations (and also linear optimizations) and in total $\mathcal{O}(\frac{G^2LD^4}{\epsilon^3})$ stochastic gradients.

## 3. The SCG algorithm

In this section, we aim to develop a stochastic variant of the Frank-Wolfe method as presented in [4] and we will refer to it as stochastic conditional gradient (SCG). This method converges to an optimal solution of (1), while it only requires access to a single stochastic gradient $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ at each iteration. Since simply replacing $\nabla F(\mathbf{x})$ with $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$ could lead the method to diverge, an averaging technique is employed to reduce the noise of the gradient approximations by recursively defining a biased estimate of the gradient as:

$$\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}(\mathbf{x}, \mathbf{z}) \tag{9}$$

with $t \in \mathbb{N}$, $\mathbf{d}_0 = 0$ and $\rho_t \to 0$ as $t$ grows, it is possible to prove that $\mathbb{E}[\|\mathbf{d}_t - \nabla F(\mathbf{x}_t)\|^2] \to 0$ asymptotically under some reasonable assumptions. This trick mitigates the high variance issue of $\nabla \tilde{F}(\mathbf{x}, \mathbf{z})$. The descent direction $\mathbf{v}_t$ is defined as the solution of the linear program

$$\mathbf{v}_t \in \underset{\mathbf{v}\in\mathcal{C}}{\operatorname{argmin}}\{\mathbf{d}_t^T \mathbf{v}\} \tag{10}$$

The algorithm pseudo-code is the following:

**Algorithm 2** Stochastic Conditional Gradient (SCG)

**Require:** Stepsizes $\rho_t > 0$ and $\gamma_t > 0$. Initialize $\mathbf{d}_0 = 0$ and choose $\mathbf{x}_0 \in \mathcal{C}$
**for** $t = 1, 2, \ldots T$ **do**
1: Update the gradient estimate $\mathbf{d}_t = (1 - \rho_t)\,\mathbf{d}_{t-1} + \rho_t \nabla \tilde{F}\,(\mathbf{x}_t, \mathbf{z}_t)$
2: Compute $\mathbf{v}_t \in \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}}\left\{\mathbf{d}_t^T \mathbf{v}\right\}$
3: Compute the updated variable $\mathbf{x}_{t+1} = (1 - \gamma_{t+1})\,\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t$
**end for**

### 3.1. SCG Convergence Analysis

The following theorem provides an upper bound for the error after $t$ iterations.

**Theorem 3.1 (SCG Convergence Rate)** *If conditions 1-4 hold, and we set $\gamma_t = 2/(t+8)$ and $\rho_t = 4/(t+8)^{\frac{2}{3}}$, then we have:*

$$\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \frac{Q}{(t+9)^{\frac{1}{3}}} \tag{11}$$

*where $Q$ is a constant*

This result shows that the expected suboptimality of the iterates generated by the SCG method converges to zero at least at a sublinear rate of $\mathcal{O}(1/t^{\frac{1}{3}})$. Therefore to achieve $\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \epsilon$ are required $\mathcal{O}(1/\epsilon^3)$ stochastic gradients and the same number of linear optimizations.

## 4. The 1-SFW algorithm

This section is dedicated to the presentation and analysis of the one sample stochastic Frank-Wolfe algorithm. For each iteration (as in SCG), 1-SFW, as the name suggests, can work with as little as one sample in order to update the optimization variable. As stated in [5] problem (1) has, in general, two formulations: the *oblivious* and the *non-oblivious* one. In the *non-oblivious* discussion the distribution $\mathbf{z} \sim P(\mathbf{z}; \mathbf{x})$ depends on the choice of $\mathbf{x}$. For the sake of semplicity we will focus just on the *oblivious* regime where the distribution $P$ is independent of $\mathbf{x}$, $\mathbf{z} \sim P(\mathbf{z})$. In stochastic convex settings, this algorithm is anyway able to achieve the best known convergence rate of $\mathcal{O}(\frac{1}{\epsilon^2})$, in terms of stochastic gradients evaluations for convex minimization, in order to reach an $\epsilon$-suboptimal solution.
This algorithms builds up on the previous results obtained in the SCG algorithm by providing an even unbiased variance reduced estimate of the gradient. To guarantee the gradient estimation to be unbiased, in the 1-SFW an unbiased estimator of the gradient variation is summed to the gradient approximation vector. This estimator is the following:

$$\Delta_t = \nabla F(\mathbf{x}_t) - \nabla F(\mathbf{x}_{t-1}) \tag{12}$$

To retrieve this estimate with only one sample $\mathbf{z}$, we basically have to calculate the difference of two consecutive stochastic gradient. So equation (12) becomes:

$$\Delta_t = \nabla \tilde{F}(\mathbf{x}_t; \mathbf{z}_t) - \nabla \tilde{F}(\mathbf{x}_{t-1}, \mathbf{z}_t) \tag{13}$$

It is important to underline that this estimate is unbiased only under oblivious settings, while in the non-oblivious case it would introduce bias.
Let's now analyse how the stochastic gradient approximation is handled in the 1-SFW algorithm. The gradient approximation is very similar to the one seen in § 2. The main difference is that $\mathbf{d}_t$ in (9) is a biased estimator of the gradient $\nabla F(\mathbf{x})$, while in 1-SFW this issue is settled adding the term $\tilde{\Delta}_t$ (unbiased estimator of the gradient variation) to $\mathbf{d}_{t-1}$. Consequently, the gradient approximation turns out to be an unbiased estimator of the gradient and its formula is the following one:

$$\mathbf{d}_t = (1 - \rho_t)(\mathbf{d}_{t-1} + \tilde{\Delta}_t) + \rho_t \nabla \tilde{F}(\mathbf{x}_t, \mathbf{z}_t) \tag{14}$$

As before, the descent direction $\mathbf{v}_t$ is defined as the solution of the linear program shown in (10).

**Algorithm 3** One-Sample SFW (1-SFW)

**Require:** Step sizes $\rho_t \in (0, 1)$ and $\gamma_t \in (0, 1)$, initial point $x_1 \in \mathcal{C}$, total number of iterations $T$
**for** $t = 1, 2, \ldots,$T **do**
1: Sample a point $z_t$ according to $p(z)$
**if** $t = 1$ **then**
2: Compute $d_1 = \nabla \tilde{F}(\mathbf{x}_1; \mathbf{z}_1)$
**else**
3: $\tilde{\Delta}_t = \nabla \tilde{F}(\mathbf{x}_t; \mathbf{z}_t) - \nabla \tilde{F}(\mathbf{x}_{t-1}, \mathbf{z}_t)$
4: $\mathbf{d}_t = (1 - \rho_t)\left(\mathbf{d}_{t-1} + \tilde{\Delta}_t\right) + \rho_t \nabla \tilde{F}\left(\mathbf{x}_t, \mathbf{z}_t\right)$
**end if**
5: Compute $\mathbf{v}_t = \arg\min_{\mathbf{v} \in \mathcal{C}} \{\mathbf{v}^T \mathbf{d}_t\}$
6: Compute the updated variable
$\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t(\mathbf{v}_t - \mathbf{x}_t)$
**end for**

### 4.1. 1-SFW Convergence Analysis

We need to display some additional assumptions needed to show the analysis of the convergence rate. First of all we assume that condition 1 (bounded constraint) is satisfied and the set $\mathcal{C}$ has diameter equal to $D = \max_{\mathbf{x},\mathbf{y} \in \mathcal{C}} \|\mathbf{x} - \mathbf{y}\|$ and radius $R = \max_{\mathbf{x} \in \mathcal{C}} \|x\|$. Then we assume

6. **Uniformly bounded function value**:

$$|\tilde{F}(\mathbf{x}, \mathbf{z})| \leq \mathbf{B} \quad \forall \mathbf{x} \in \mathcal{C} \quad \text{and} \quad \forall \mathbf{z} \in \mathcal{C} \tag{15}$$

Moreover the function $\tilde{F}(\mathbf{x}, \mathbf{z})$ has to be $L$-smooth. Under these assumptions we can state that:

**Theorem 4.1** *If $F$ is convex and we assign $\rho_t = (t-1)^{-1}$ and $\gamma_t = t^{-1}$, we obtain that $x_{t+1} \in \mathcal{C}$ is feasible and satisfies*

$$\mathbb{E}[F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*)] \leq \mathcal{O}(t^{-1/2}) \tag{16}$$

Through this theorem, it is possible to see that 1-SFW only needs at most $\mathcal{O}(\frac{1}{\epsilon^2})$ stochastic oracle queries and linear optimizations to reach $\epsilon$-suboptimal solution for convex minimization.

## 5. The SVRF algorithm

(*Premise: we denote that the analysis and implementation of this algorithm was not required for the project, thus the work presented in this section is completely additional. It is also denoted that this is the only algorithm analyzed which on purpose computes a full gradient every so often to improve the method, thus an equal ground comparison with the previous algorithms would be unfair).*
In the stochastic variance-reduced Frank-Wolfe, the exact gradient is replaced with the average of a mini-batch of

variance-reduced stochastic gradients at some point $\mathbf{w} \in \Omega$ with some snapshot $\mathbf{w}_0 \in \Omega$, which is described as

$$\tilde{\nabla} f(\mathbf{w}; \mathbf{w}_0) = \nabla f_i(\mathbf{w}) - (\nabla f_i(\mathbf{w}_0) - \nabla f(\mathbf{w}_0)) \quad (17)$$

where *i* is chosen uniformly at random from $\{1, ....n\}$ and the term $\nabla f_i(\mathbf{w}_0) - \nabla f(\mathbf{w}_0)$ is necessary, as a correction term, to reduce the variance of the stochastic gradient. An important element of this algorithm is the snapshot $\mathbf{w}_0 \in \Omega$, which is typically a decision point from some previous iteration whose exact gradient is pre-computed.

**Algorithm 4** Stochastic Variance-Reduced Frank-Wolfe (SVRF)

**Require:** Parameters $\gamma_k$, $m_k$ and $N_t$
1: **Initialize:** $\mathbf{w}_0 = \min_{\mathbf{w} \in \Omega} \nabla F(\mathbf{x})^T \mathbf{w}$ for some arbitrary $\mathbf{x} \in \Omega$
**for** $t = 1, 2, \ldots T$ **do**
2: Take snapshot $\mathbf{x}_0 = \mathbf{w}_{t-1}$ and compute $\nabla F(\mathbf{x}_0)$
**for** $k = 1$ **to** $N_t$ **do**
3: Compute $\tilde{\nabla}_k$, the average of $m_k$ iid samples of $\tilde{\nabla} f(\mathbf{x}_{k-1}, \mathbf{x}_0)$
4: Compute $\mathbf{v}_k = \min_{\mathbf{v} \in \Omega} \tilde{\nabla}_k^T \mathbf{v}$
5: Compute $\mathbf{x}_k = (1 - \gamma_k)\, \mathbf{x}_{k-1} + \gamma_k \mathbf{v}_k$
**end for**
Set $\mathbf{w}_t = \mathbf{x}_{N_t}$
**end for**

### 5.1. SVRF Convergence Analysis

**Theorem 5.1** *If conditions 1-3 hold and setting:*

$$\gamma_t = \frac{2}{t+1}, \quad m_k = 96(t+1), \quad N_t = 2^{t+3} - 2$$

*the SVRF ensures* $\mathbb{E}[F(\mathbf{w}_t) - F(\mathbf{w}^*)] \leq \frac{LD^2}{2^{t+1}}$

This theorem can be proved by induction. Assuming it to be true we can deduce that to achieve a $1 - \epsilon$ accuracy the iterations' number of the first for cycle that goes up to $T$, should be of order $\Theta(\log_2 \frac{LD^2}{\epsilon})$. Now looking at the algorithm we see that step 2 is done $T + 1$ times, step 3 is done $\sum_{t=1}^{T} \sum_{k=1}^{N_t} m_k = \mathcal{O}(4^T)$ times and step 5 requires $\sum_{t=1}^{T} N_t = \mathcal{O}(2^T)$ iterations. Putting together these things we deduce that the SVRF requires $\mathcal{O}(\log_2 \frac{LD^2}{\epsilon})$ exact gradient evaluations, $\mathcal{O}(\frac{L^2 D^4}{\epsilon^2})$ stochastic gradient evaluations and $\mathcal{O}(\frac{LD^2}{\epsilon})$ linear optimizations.

## 6. Application

### 6.1. Problem: Logistic Regression

In order to support what was reported in the previous theoretical analysis, we are going to show the results achieved through some experiments, comparing the performances of the various algorithms. We chose to evaluate the algorithms on the *logistic regression* problem, more specifically the $l_1$-constrained logistic regression, since the constraint set is of the form of the $l_1$ ball: $\mathcal{C} = \left\{\mathbf{W} \in \mathbb{R}^{\mathbf{d} \times C} : \|\mathbf{W}\|_1 \leq r\right\}$, for some constant $r \in \mathbb{R}_+$. The aim is to minimize the loss function of the logistic regression over $\mathcal{C}$

$$F(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{W}_c^{\mathbf{T}} \mathbf{x}_i\right)\right) \quad (18)$$

where each data point *i* is indicated by the pair $(\mathbf{x}_i, y_i) \in \mathbb{R}^d$, with $\mathbf{x}_i$ is the feature vector and $y_i \in \{1, ..., C\}$ is the corresponding label. We will deal with binary classification so $y_i \in \{+1, -1\}$. We remark that $\|\mathbf{W}\|_1$ is the $l_1$ norm. In the general case $\|\mathbf{W}\|_1 = \max_{1 \leq j \leq \mathcal{C}} \{\sum_{i=1}^{d} |[\mathbf{W}_{ij}]|\}$. The function $F(\mathbf{W})$ is convex and smooth.
The full gradient of the logistic regression function is the following:

$$\nabla F(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y^i \cdot \mathbf{x}_i}{1 + \exp\left(y_i \cdot \mathbf{W}^{\mathbf{T}} \mathbf{x}_i\right)} \quad (19)$$

Thus the *j*-th component of the gradient is

$$\frac{\partial F(\mathbf{W})}{\partial \mathbf{W}_j} = \frac{1}{n} \sum_{i=1}^{n} \frac{-y^i \cdot x_i^j}{1 + \exp\left(y_i \cdot \mathbf{W}^{\mathbf{T}} \mathbf{x}_i\right)} \quad (20)$$

### 6.2. Dataset

The performances of the algorithms are evaluated on two different datasets, both related to a problem of binary classification. The first dataset is *"Heart disease prediction"* ([2]), which presents over 4,000 records and 15 attributes to predict if a subject is going to have or not a heart disease. The second one is *"Diabetics prediction"* ([1]) characterized by 8 features and almost 800 records and it is used to predict if a person is having diabetics or not.

### 6.3. Implementation

As already mentioned before, we are dealing with a problem of the form $l_1$ Ball, since our constrain set is $\mathcal{C} = \left\{\mathbf{W} \in \mathbb{R}^{\mathbf{d} \times C} : \|\mathbf{W}\|_1 \leq r\right\}$. So in this case the solution we adopted to solve the Frank-Wolfe problem is the following one:

$$\mathbf{v}_t = sign(-\nabla_{i_t} f(x_t)) \cdot e_{i_t} \quad (21)$$

with $i_t = \underset{i}{\operatorname{argmax}} |\nabla_i f(x_t)|$. In the case of SFW and SVRF, this solution is computed with the stochastic gradient, chosen randomly, while in the other algorithms it is used the estimate of the gradient $\mathbf{d}_t$.
As stopping condition we use the maximum number of iterations, while the starting point is a feasible point belonging to the constraint.

## 7. Results

Now we report the results obtained running the algorithms to optimize $F(\mathbf{W})$ using the datasets mentioned above. A fundamental premise has to be made before presenting the results. In our experiments we considered two distinctly different scenarios:

1. In the first case we will respect the theoretical assumptions made by SFW, which require a growing mini batch size of $t^2$ samples to obtain the theoretical results presented, while the other two algorithms considered (SCG and 1-SFW) will use a single sample per gradient descent step. This implies that, after a specific number of iterations, the SFW algorithm will develop into a deterministic algorithm that will utilize the whole dataset to perform each gradient descent step. Therefore that's why it will perform better than its still stochastic counterparts SCG and 1-SFW at equal iterations or computation time.

2. In the second case, to have an equal algorithm comparison setting, we will consider the SFW algorithm with a constant batch size of 1 sample, which is the same batch size used in the other two algorithms. However, this will **NOT** respect the assumptions made by the paper of quadratic batch size. Nevertheless this setting was implemented to have a comparison between the three algorithms where each of them operates in a stochastic manner.

### 7.1. Diabetes Dataset

#### 7.1.1 SFW ($t^2$), SCG, 1-SFW

In figure 1 we see the loss decreasing with respect to the number of iterations using the diabetes dataset. We fixed the number of iterations for the SFW algorithm to 100 while for the other two to 800. As we can see by the plot, the SFW algorithm is performing better than its competitors with much less iterations due to the fact that (as previously stated), it will perform a deterministic gradient descent and not anymore a stochastic gradient descent after a few iterations; which of course performs better (iteration-wise) than its stochastic counterpart.

In figure 2 we see the CPU time required by the algorithms. The SFW algorithm, takes longer than the others because, even though the number of iterations is fixed to be lower than its counterparts, a full deterministic gradient descent step takes much longer than a stochastic one.

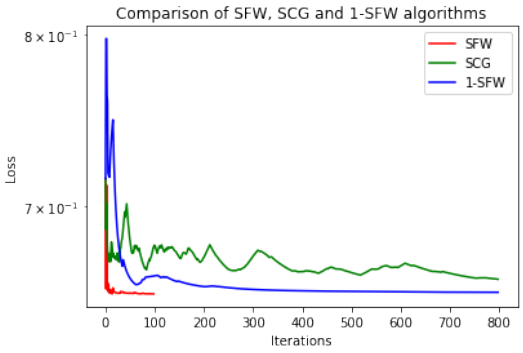


Figure 1: Decreasing loss function vs iterations with the diabetes dataset, SFW minibatch $\sim t^2$

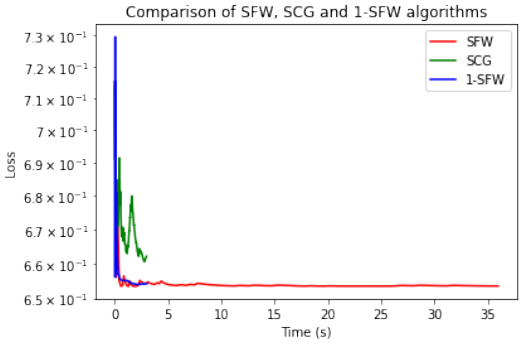


Figure 2: Decreasing loss function vs CPU time in seconds with the diabetes dataset, SFW minibatch $\sim t^2$

#### 7.1.2 One sample batch size

Now we consider the same analysis using a one sample batch size for all algorithms. As shown in figure 3 (w.r.t. iterations) and in figure 4 (w.r.t. computation time), now we have that the 1-SFW algorithm achieves the best results.

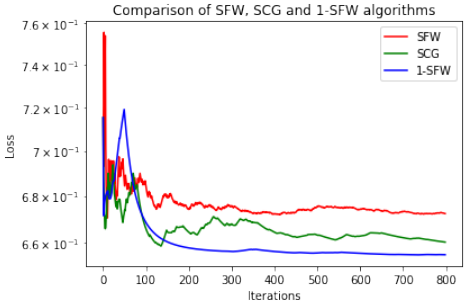


Figure 3: Decreasing loss function vs iterations with the diabetes dataset, one sample batch

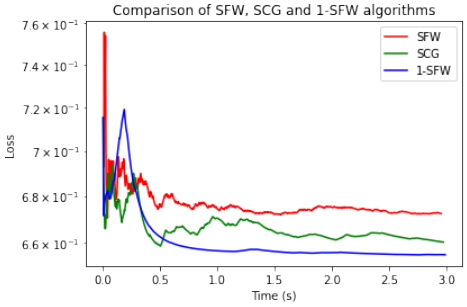


Figure 4: Decreasing loss function vs CPU time in seconds with the diabetes dataset, one sample batch

#### 7.1.3 Expected noise of the gradient estimation

For the SCG algorithm we made also an empirical proof of the fact that $\mathbb{E}[\|\mathbf{d}_t - \nabla F(\mathbf{x}_t)\|^2] \to 0$. For the diabetes dataset in fact we see (figure 5) that the statement is respected.

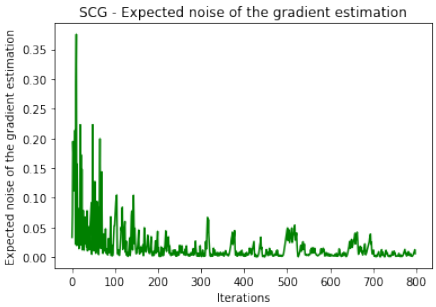


Figure 5: SCG Expected noise, diabetes

### 7.2. Heart Dataset

#### 7.2.1 SFW ($t^2$), SCG, 1-SFW

The same analysis (with a growing mini batch size for SFW) is carried out with the heart disease dataset, obtaining figures 6 and 7.

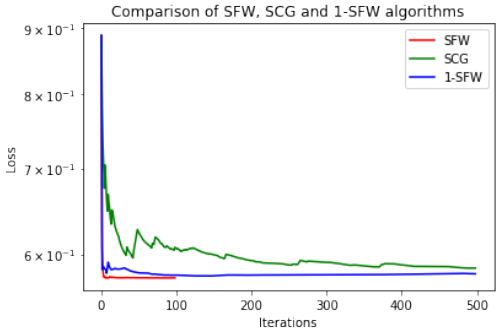


Figure 6: Decreasing loss function vs iterations with the heart dataset, SFW minibatch $\sim t^2$

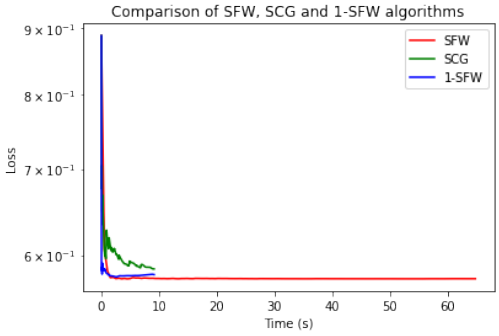


Figure 7: Decreasing loss function vs CPU time in seconds with the heart dataset, SFW minibatch $\sim t^2$

#### 7.2.2 One sample batch size

The results are similar to the diabetes ones also for what concerns the one sample minibatch. With the results being shown in plot 8 and in 9.

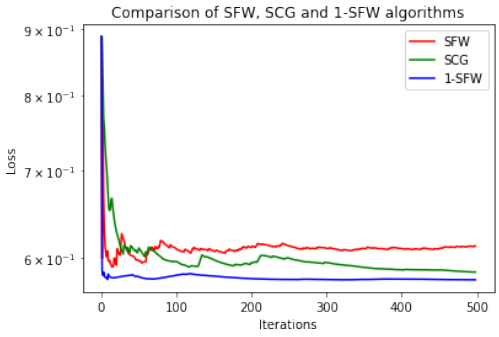


Figure 8: Decreasing loss function vs iterations with the heart dataset, one sample batch

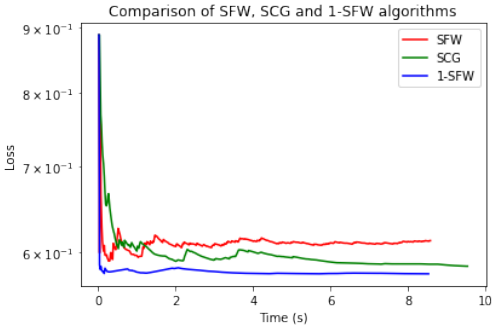


Figure 9: Decreasing loss function vs CPU time in seconds with the heart dataset, one sample batch

### 7.2.3 Expected noise of the gradient estimation

Also with this dataset we prove empirically the fact that $d_t$ for the SCG converges in expectation to the exact gradient as shown in figure 10.

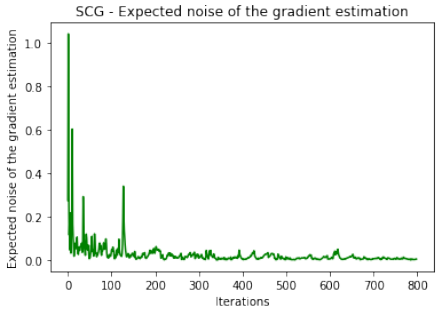


Figure 10: SCG Expected noise, heart

## 8. SVRF on Diabetes and Heart datasets

As last thing we show how the SVRF performs over the previous two datasets. For the diabetes dataset we have:

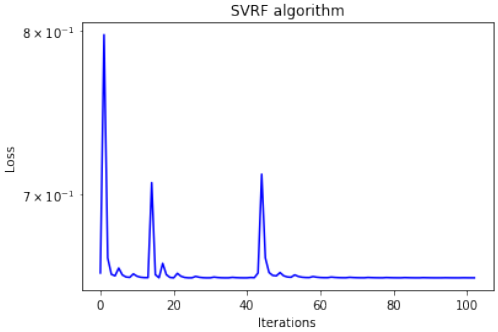


Figure 11: Decreasing loss function vs iterations with the diabetes dataset

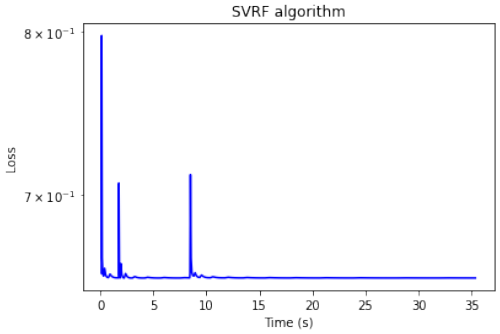


Figure 12: Decreasing loss function vs CPU time in seconds with the diabetes dataset

For the heart dataset we have:

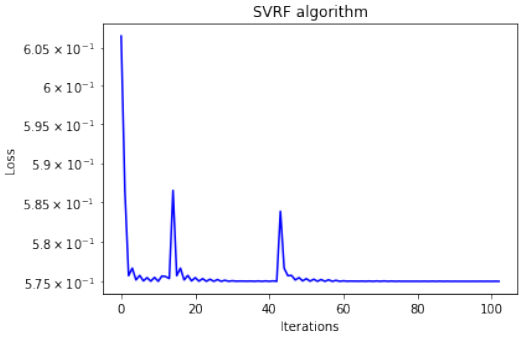


Figure 13: Decreasing loss function vs iterations with the heart dataset

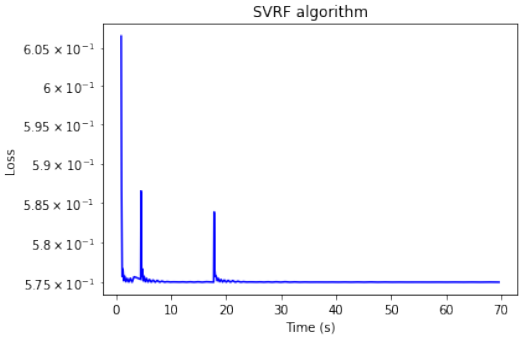


Figure 14: Decreasing loss function vs CPU time in seconds with the heart dataset

## References


[1] Diabetics prediction using logistic regression. https://www.kaggle.com/datasets/kandij/diabetes-dataset.

[2] Logistic regression To predict heart disease. https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression.

[3] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. *CoRR*, abs/1602.02101, 2016.

[4] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization, 2018.

[5] Mingrui Zhang, Zebang Shen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. One sample stochastic frank-wolfe, 2019.