Frank-Wolfe for Coding

Optimization For Data Science

Prof. Francesco Rinaldi

BERKE FURKAN KUSMENOGLU - 2041496
NILSU OZ - 2041502

June 25, 2022

1 Introduction

We analyze stochastic optimization problems for a broad class of objective functions, including convex and continuous submodular functions. However, for large problems, the application of stochastic proximal gradient methods remains limited because for a convex set projection to be computationally efficient, the problem dimension must be very large. It is possible to optimize an objective function defined as an expectation of a collection of random functions subject to a general convex constraint in this class of problems. Here we consider an optimization variable x * X * Rn and a random variable Z * Z that together determine a stochastic function F: X × Z → R. The objective is to solve the problem

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

Figure 1:

The article explains how to learn a new orthogonal dictionary dynamically from streaming data without storing any historical data using an online orthogonal dictionary. The main challenge is to solve problem without the accessibility of F(D) or ∇F(D) An efficient online algorithm with convergence analysis is designed with a novel problem formulation. Two algorithms discussed in this paper are: Stochastic Frank Wolfe (SFW) and NonConvex Frank Wolfe Method. This paper will address the methodology of the two papers [1,2], compare their theoretical differences and prove those differences through an empirical implementation and analysis using these algorithm To enable an efficient online algorithm, we relax the orthogonal constraint in the problem formulation. The proposed online ODL scheme has demonstrated its effectiveness and efficiency by using synthetic data and real-world MNIST handwriting digit data.

2 Algorithms

2.1 Stochastic Frank Wolfe Method

In this part we will be explaining steps we follow to implied Stochastic Frank Wolfe Method. Suppose each iteration yields an unbiased estimation of the stochastic gradient *F(x), with each iteration providing the stochastic gradient F(x). It is well known that a naive stochastic implementation of Frank-Wolfe may diverge due to the nonvanishing variance of the gradient approximations. To address this problem, we introduce a stochastic version of the Frank-Wolfe algorithm that reduces the noise of the gradient approximations using an averaging technique commonly used in stochastic optimization.

For our Stochastic Frank-Wolfe Algorithm we defined the biased gradient estimation. In

Figure 2, to define our stepsize we use the reference from article[1] to choose best performance for stepsizes. Purpose of the stepsize is, it needs to be get close to zero when the t grows.

$$G_t = (1 - p_t)G_{t-1} + \frac{p_t}{M_t}\sum_{j \in [M_t]} -\nabla \left\| D_{t-1}^T\, y_t^j \right\|_3^3$$

Figure 2:

Our expectation was to prove that, for approximating the gradient, a biased gradient estimate is a better choice when we compare it with the unbiased gradient estimate. In the next step, we describe the v_t which is our descent direction in Figure 3. We want to minimize the transpose of multiplication of gradient approximation and v. For that, we use Singular Value Decomposition as $U, \Sigma, V^T = SVD(d_t^T v)$. So the desired argmin is: $v_t = U.V^T$

$$\mathbf{v}_t \in \underset{\mathbf{v} \in C}{\operatorname{argmin}}\{\mathbf{d}_t^T \mathbf{v}\}.$$

Figure 3:

Lastly, we update the dictionary with the stepsize and the descent direction in every iteration.

$$\mathrm{d}_{t+1} = (1 - \gamma_{t+1})\mathrm{x}_t + \gamma_{t+1}\mathrm{v}_t$$

Figure 4:

2.2 NonConvex Stochastic Frank Wolfe Method

In the second paper[2], we analyzed a Frank-Wolfe-based algorithm, the Nonconvex Stochastic Frank-Wolfe Method, to solve the relaxed problem directly without the problem transformation. can achieve low-complexity computation per iteration due to the linear minimization oracle in the Frank-Wolfe method. In this section, we first introduce the proposed Frank-Wolfe-based algorithm, NoncvxSFW, for general nonconvex online problems with convex constraints, and then specialize it to solving the problem in Figure 5.

$$\rho = \frac{\text{minimize}}{\mathrm{D} \in \beta_{\mathrm{SP}}(\mathrm{N}, \mathrm{R})} \mathrm{F}(\mathrm{D}) = \mathrm{E}_{\mathrm{y} \sim \mathrm{p}}[-\|\mathrm{D}^t \mathrm{y}\|_3^3]$$

Figure 5:

```
**Data:** $\{Y_t\}_{t=1}^{\infty}$ with $Y_t = [y_t^1, \dots, y_t^{M_t}]$
**Result:** $\{D_t\}_{t=1}^{\infty}$
Initialization: $G_0 = 0$ and random
 $D_0 \in \mathbb{O}(N, \mathbb{R}) \subset \mathbb{B}_{sp}(N, \mathbb{R})$
**for** $t = 1, 2, \dots$ **do**
    $\rho_t = 4(t+1)^{-1/2}$, $\gamma_t = 2(t+2)^{-3/4}$
    1. Gradient Approximation:
     $G_t = (1 - \rho_t) G_{t-1} + \frac{\rho_t}{M_t} \sum_{j \in [M_t]} -\nabla \| D_{t-1}^{\mathrm{T}} y_t^j \|_3^3$
    2. LMO: $U, \Sigma, V^{\mathrm{T}} = \mathrm{SVD}(-G_t)$
            $S_t = U V^{\mathrm{T}}$
    3. Variable Update:
    $D_t = Polar((1 - \gamma_t) D_{t-1} + \gamma_t S_t)$.
```

Figure 6:

First we initialize G_0 as 0 and choose a random D_0 from orthogonal group by using the scipy.stats library.

Step 1 (Gradient Approximation): For the sampled gradient, we used the expression that is given in the paper.

$$-\nabla \| D^T y_t^j \|_3^3 = -y_t^j (|(D^{(t-1)})^{\mathrm{T}} y_t^j| \odot (D^{(t-1)})^{\mathrm{T}} y_t^j)^{\mathrm{T}}$$

Figure 7:

Step 2 (LMO): To achieve argmin of the <G_t,S> where S $\in$ B(N,R) unit spectral ball we use the Singular Value Decomposition. The desired minimum is UV^T which is from SVD(-G). However it can be also achieved by the polar decomposition such as $P = V\Sigma V^T$ and $U = WV^T$ where $SVD(A) = W\Sigma V^T$. We use numpy.linalg module to compute SVD.

Step 3 (Variable Update): At the end we adopt polar decomposition by using scipy.linalg module. From the polar decomposition, we get two returns: P (positive semi-definite Hermitian matrix) and U (unitary matrix). We used the matrix U which has an orthonormal column. For further confirmation, we checked the Lemma 4. We can see that $Polar(X) \in B(N, R)$ and $F(Polar(X)) \leq F(X)$

3 Experimentes

In this section, we will be calculating the effectiveness and efficiency of our methods without Synthetic Data and the Real-World MNIST dataset. All of the experiments are executed in Python 3.8 with a 3.10-GHz Apple M1 processor. In Y.Xue's work[2] there are four baseline methods. For our project, we did work on two baseline methods.

3.1 Baseline 1 - Stochastic Frank-Wolfe Method

To address the online ODL problem in Figure 5, this baseline uses the recently proposed Stochastic Frank-Wolfe algorithm based on paper [2]. We compare the proposed NoncvxSFW algorithm to this baseline in terms of convergence property to demonstrate its effectiveness and efficiency in addressing the online ODL problem P.

3.2 Baseline 2 - NonConvex Stochastic Frank-Wolfe Method

This baseline uses the NonConvex Stochastic Frank-Wolfe Method based on paper [1]. From the paper we understand that using $-||.||_3^3$ has an advantage. So we use the negative $l_3 - norm$

3.3 Datasets

3.3.1 Synthetic Data

We generate our Synthetic Data through 100 independent Monte Carlo trials. For each trial l, we choose our ground-truth dictionary $D^{true}(l)$ from the orthogonal group by using scipy.linalg module and calculate QR Decomposition which gives us Q (NxN matrix). Then we choose sparse signals $x_t^j \in \mathbb{R}^N$ $(j \in [M_t], 1 \le t \le T)$ from Bernoulli-Gaussian distribution. Then we calculate $y_t^j = D^{true}(l)x_t^j$ All of the methods, we use the same random starting point in each trial to ensure a fair comparison. For data generation, we used $M_t = B_t$ which we fixed through all iterations, and $T = 3 * 10^3$ without sacrificing generality. The error measure at time index t is calculated to assess the convergence property.

$$\text{Error}_t = \frac{1}{100}\sum_{l=1}^{100}\left| 1 - \frac{||D_t^T(l)D^{true}(l)||_4^4}{N} \right|$$

Figure 8:

Since our dictionary D belongs to the orthogonal group, we know that $D^T D$ will give us the identity matrix. If the algorithm works perfectly we expect D_t that we get from the algorithm will be exactly the same as the ground truth D^{true}. So in the best-case scenario, ${D_t}^T D^{true}$ will return the identity matrix. Taking the p^{th} power of the l_p norm basically means the summation of every element's p^{th} power. Hence taking the 4^{th} power of the elements in the identity matrix and summing them together will give us N*1. And dividing this result by N is 1. Thus we proved that for the best-case scenario the error will reach zero.

Here is the graph for N = 10, B = 10, θ = 0.3.

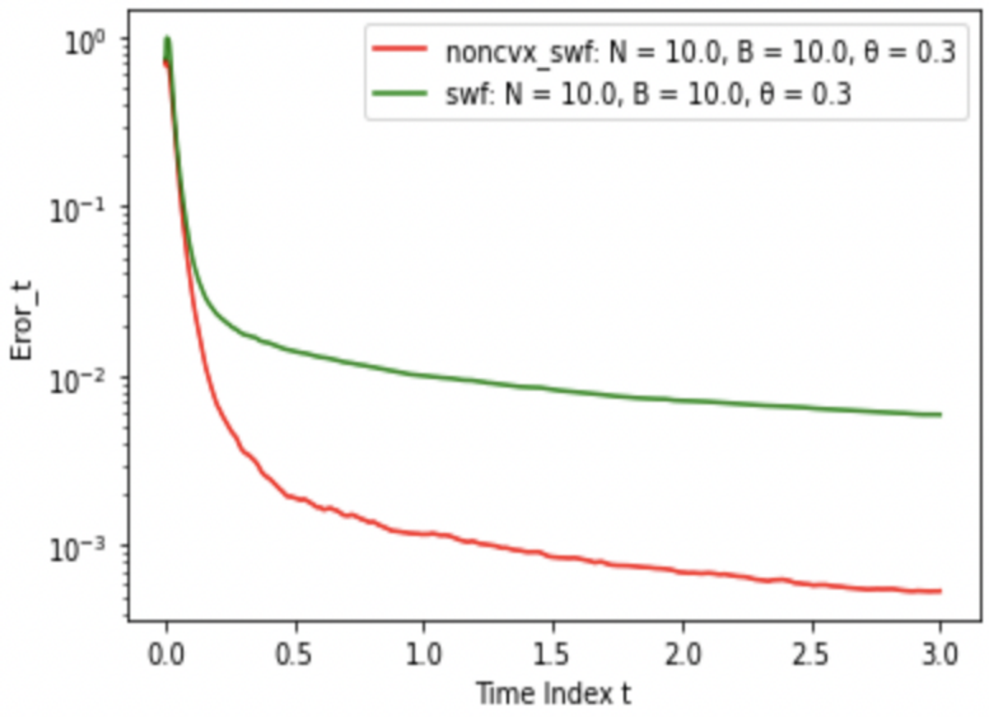


Figure 9:

We tried different parameters on the algorithm to see the impact of θ and B. The graphs of the different minibatch sizes:

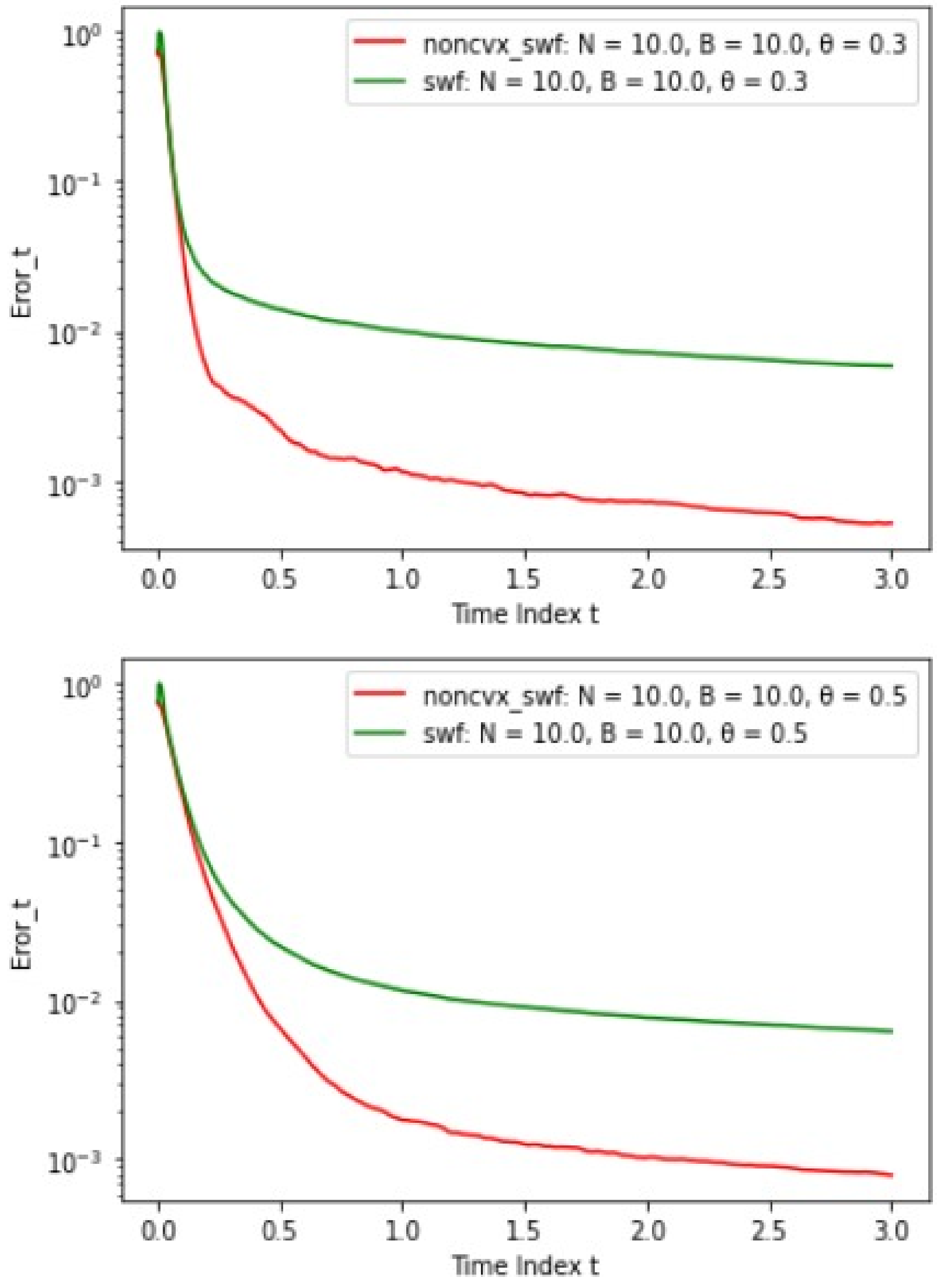
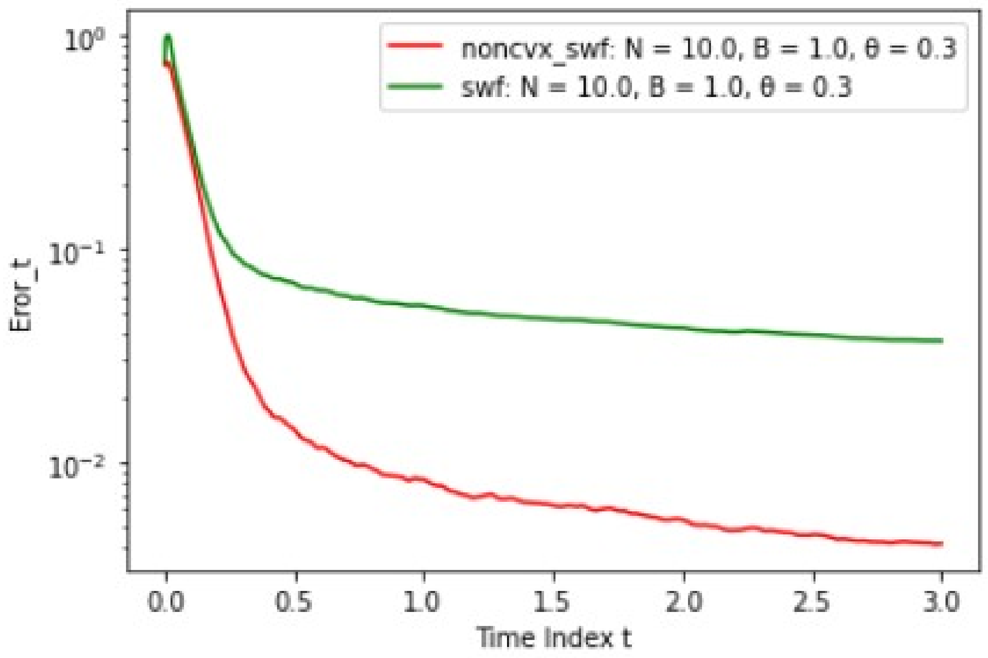


Figure 10:

We can see that when the batch size gets larger, the algorithms can converge faster and better.

The graphs of the different θ values:



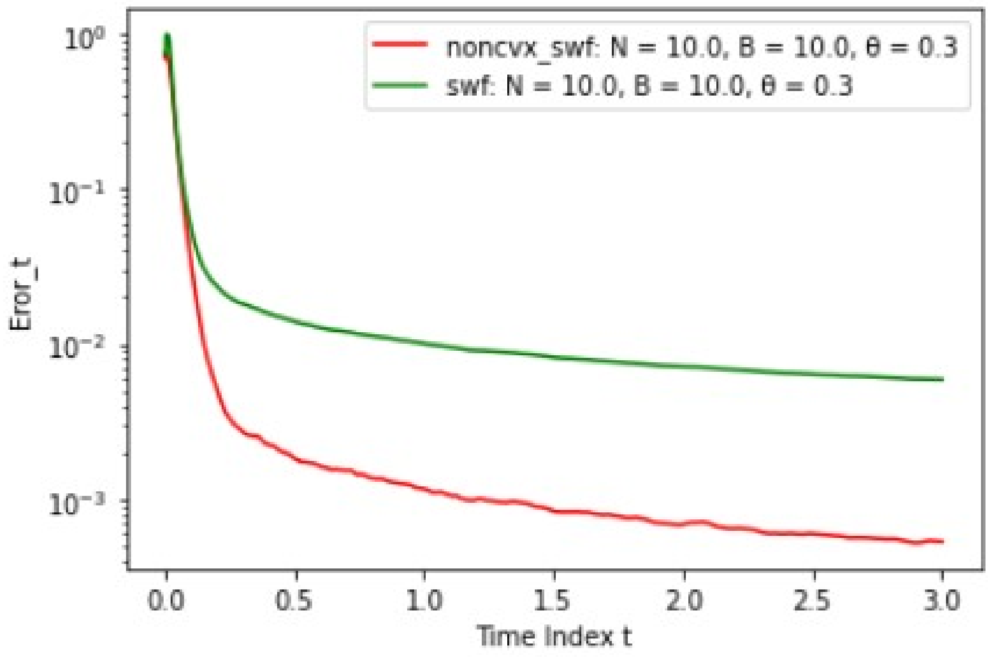


Figure 11:

We can see that with smaller θ the algorithms has faster convergence.To sum up the experiments,in the all experiments we imply, the Nonconvex Stochastic Frank Wolfe converge better

3.3.2 Real-World Dataset

For a real world dataset we use MNIST handwriting digits. It consists of 60000 training and 10000 test with the greyscale which has a size of 28x28 pixels.

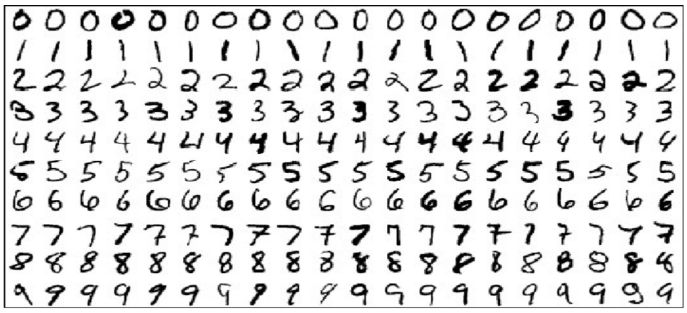


Figure 12:

Data Set Preparation: We choose 2000 samples for the MNIST dataset. Then we choose minibatchsize as 10. So we split the 2000 samples into 200 samples which has a size of 784x10. then we implement T=200 iterations on the two algorithms and achieved D_t from both of them. Then we calculate the error.

For the error calculation we used the metrics we tried was RMSE (Root-mean-square-deviation). Which we define as in the Figure 13.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=0}^{T} \sum_{j=1}^{M_t} \left\| \tilde{y}_t^j - y_t^j \right\|_2^2}{\sum_{t=0}^{T} \sum_{j=1}^{M_t} \left\| y_t^j \right\|}}$$

Figure 13:

The RMSE scores from Nonconvex Stochastic Frank Wolfe method with respect to different n_0 which is the number of non zero values in the sparse code:

Nonconvex SFW (n_0)	RMSE Scores(%)
28	4.83
14	1.78
7	0.69
2	0.19

For the Stochastic Frank Wolfe method we couldn't achieve good results. The problem may be the construction of the mini-batches, the size of the dictionary or this method may not be a good choice of this dataset.

4 Conclusion

As part of our research, we use stochastic Frank-Wolfe and NonConvex-Stochastic Frank-Wolfe to solve convex minimization and submodular maximization problems. Experiments with synthetic and real-world data were conducted to verify the correctness of our theoretical predictions and demonstrate the superior performance of our proposed method over baselines. A number of machine learning and optimization communities are increasingly interested in Frank-Wolfe methods (in the convex case) because of their projection-free property and their ability to exploit structured constraints. Despite this, we are relatively unaware of how these algorithms work on a nonconvex problem. We propose nonconvex stochastic Frank-Wolfe and stochastic Frank-Wolfe methods in this paper and analyze their convergence properties. Finally, we show that in the nonconvex problem NonConvex-Stochastic Frank-Wolfe shows better convergence better than Stochastic Frank-Wolfe.

5 References

[1] Y. Xue and V. K. N. Lau, ”Online Orthogonal Dictionary Learning Based on Frank-Wolfe Method,” in IEEE Transactions on Neural Networks and Learning Systems

[2] Mokhtari, Aryan Hassani, Hamed Karbasi, Amin. (2018). Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization.

[3] Shen, Yifei Xue, Ye Zhang, Jun Letaief, Khaled Lau, Vincent. (2020). Complete Dictionary Learning via ℓ_p-norm Maximization.

[4]Grossi G, Lanzarotti R, Lin J. Orthogonal Procrustes Analysis for Dictionary Learning in Sparse Linear Representation.

[5]Xue, Ye Shen, Yifei Lau, Vincent Zhang, Jun Letaief, Khaled. (2020). Blind Data Detection in Massive MIMO via -Norm Maximization Over the Stiefel Manifold

[6] Q. Qiu, V. M. Patel and R. Chellappa.(2014) ”Information-Theoretic Dictionary Learning for Image Classification,”

[7] Xingyao Ye and A. Yuille.(2011)”Learning a dictionary of deformable patches using GPUs,”