One-Layer Neural Network Training with Frank-Wolfe

Mojtaba Amini[1] and Kamile Dementaviciute[2]

[1]Department of Mathematics, University of Padua
[2]Department of Computer Science, Vilnius University

August 2022

Abstract

In this paper we study Frank-Wolfe for a simple neural network. Frank-Wolfe method is a useful optimization technique for constrained problems in machine learning because of its projection free property. We draw a comparison among different stochastic version of Frank-Wolfe method which is implemented for a one-layer neural network. We compare the results of Stochastic Frank-Wolfe(SFW) and Stochastic Variance-Reduced Frank-Wolfe(SVRF) for a convex constraint problem on an l_1- ball region. The results are presented having trained the models on two binary classification datasets. The results show that the variance reduction technique improves the convergence rate at a higher computational cost, however it requires fewer iterations.

1 Introduction

Today, neural networks are one of the most commonly used machine learning techniques. Neural networks optimize finite-sum problems. The finite sum function is(1):

$$\min_{\theta\in\Omega} F(\theta) = \min_{\theta\in\Omega} \frac{1}{m}\sum_{i=0}^{m} f_i(\theta) \tag{1}$$

We assume that F, f_i ($i \in 1, 2, ..., m$) are all smooth, differentiable, and that the Ω is a convex set but F, f_i possibly non-convex. Since in this study we will be working with binary classification tasks, the cost function is as follows(2):

$$\min_{\theta\in\Omega} F(\theta) = -\frac{1}{m}\sum_{i=0}^{m}(\theta^i log(\hat{\theta}^i) + (1-\theta^i) log(1-\hat{\theta}^i)) \tag{2}$$

Solving the above equation by one of the most popular methods in NN when dealing with a large datasets like SGD (stochastic gradient descent), requires the gradient descent calculations for each sample or mini-batch. In search of improving optimization methods, regularization's technique has been used in many cases. Motivated by this approach, the efficiency of the neural network on a constrained domain has been considered[4]. On the other hand, implementing any gradient descent method for a constrained parameter space to a convex compact region requires one additional step at each iteration called projection, which in the case of SGD can be extremely costly[4]. This step ensures that the parameters remain in the constrained domain, however it increases the computational cost of the methods. For this reason, the need of projection free methods like Frank-Wolfe to deal with constrained optimization problems arises. In this study, we will explore the the application of stochastic variants of Frank-Wolfe methods.

$$v_t = \arg\min_{v\in c} \langle \tilde{\nabla}(J_t, v)\rangle \tag{3}$$

The FW methods assumes access to a linear minimization oracle (LMO)(3) to ensure that v_t moves in this descent direction and find the update value within the constrained region (4). In the LMO problems, if we consider a polytope convex region which is consist of a set of vertices, the result of the LMO assumed to be one of the vertices of this convex region.

$$\theta_{t+1} = \theta_t + \alpha(v_t - \theta_t) \tag{4}$$

In this study, we are going to compare the stochastic FW on a l_1-ball feasible set1., which is $C(radius) = \{x \in R^n : ||x||_1 \leq radius\}$, which the diameter of this polytope is $diameter = 2 \times radius$. Also the result of the LMO is:

$$v_t = \begin{cases} diameter \times sign(-\nabla_{i_k} f(\theta_k))\cdot, & \text{if } i_k = \arg\max_i |\nabla_i f(\theta_k)|. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The result of the (5) is a sparse vector with a non-zero element.

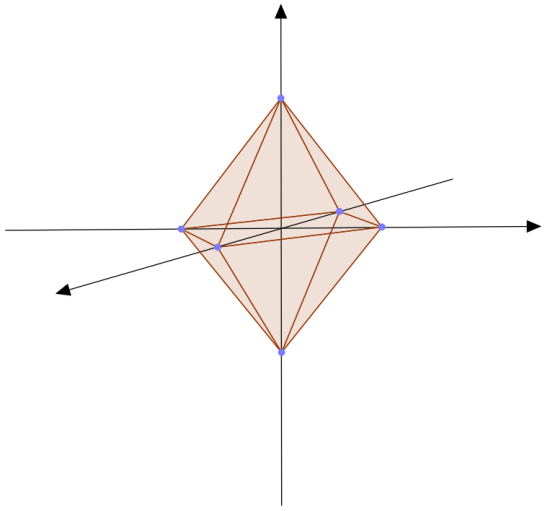

Figure 1: $l_1 - ball$[3]

The pseudo-code related to Frank-Wolfe algorithm with l_1-ball is

Algorithm 1 Frank-Wolfe method for l_1-ball

Require: Starting from a point inside the region
  for k=1,.... **do**
    Set $\hat{\theta}_k = diameter \times sign(-\nabla_{i_k} f(\theta_k))$, with $i_k = \arg\max_i |\nabla_i f(\theta_k)|$
    if $\hat{x}_k$ satisfies some specific condition, then STOP
    Set $\theta_{k+1} = \theta_k + \alpha_k(\hat{\theta}_k - \theta_k)$
  end for

Our contribution We have studied the stochastic variants of FW for a one-layer Neural Network when the constraint is $l_1 - ball$, and have compared the results on two datasets. In the next chapter we will go into a more detailed explanation of these Stochastic Frank-Wolfe methods. Before that we will shortly evaluate the given papers.

Related Work In the paper that we are referencing the most for our experiments, Sebastian Pokutta et al. [4] examines the empirical efficacy of using stochastic Frank–Wolfe methods like SFW, MSFW and SVRF for training Neural Networks with different constrained parameters. In another study, Hazan, et al, work [2] used recent variance reduction technique, stochastic version of frank-wolfe. Hazan, et al showed that the variance reduction technique, previously shown to be highly useful for gradient descent variants, can also be very helpful in speeding up projection-free algorithms. Moreover, in Sashank J Reddi,et al, work [5] Frank-Wolfe methods for non-convex stochastic and finite-sum optimization problems are evaluated. The convergence rates in this paper were similar to those obtained for online Frank-Wolfe and only slightly worse than those obtained for stochastic Frank-Wolfe in the convex setting. Xie,et al [6] studied the projection-free methods for for solving smooth Online Convex Optimization (OCO) problems is evaluated. These methods achieve optimal regret bounds while possessing low per-iteration computational costs. The proposed methods achieve nearly optimal regret bounds with low computational costs.

2 Stochastic Variants of Frank-Wolfe

Calculating the whole gradient at each iteration, when we are dealing with a large number of data points needs a large amount of memory and computational time. Therefore, using stochastic version of FW is proposed, which requires only the stochastic gradient calculation instead of the full gradient, and we just need to determine argmax of a vector to find the right direction to move. To achieve $1-\epsilon$ accuracy, we have compared the number of gradients calculated during the optimization in table1.

Table 1: Comparisons of different Frank-Wolfe variants[2].

Algorithm	Extra Conditions	Exact Gradients	Stochastic Gradients
FW	G-Lipschitz	$\mathcal{O}(\frac{LD^2}{\epsilon})$	0
SFW	-	0	$\mathcal{O}(\frac{G^2LD^4}{\epsilon^3})$
SVRF	-	$\mathcal{O}(\frac{LD^2}{\epsilon})$	$\mathcal{O}(\frac{L^2D^4}{\epsilon^2})$

L is Lipschitz constant, D is the feasible set diameter $||x-y|| \leq D$ for every $x, y \in \Omega$.

2.1 Stochastic Frank-Wolfe (SFW)

Stochastic version of FW calculates the gradient only for a set of random samples in the mini-batches, and the rest of the algorithm is the same as before. In [4], it has been proved that $\tilde{\nabla} L(\theta_k)$ is an unbiased estimator for total gradient matrix in the simple FW, and that the new point $\theta_{k+1}$is still in convex set[3]. The stochastic version of the algorithm without considering a momentum term is provided in the algorithm below.

Algorithm 2 Stochastic Frank-Wolfe method for l_1-ball

Require: Starting from a point inside the region
  for k=1,.... **do**
    Uniformly sample i.i.d. $i_1.i_2, ..., i_b$ from $[1, .., n]$
    $\tilde{\nabla} L(\theta_k) \leftarrow \frac{1}{b} \sum_{j=1}^{b} \nabla f_{i_j}(\theta_k)$
    Set $\hat{\theta}_k = diameter \times sign(-\tilde{\nabla}_{i_k} L(\theta_k))$, with $i_k = \arg\max_i |\tilde{\nabla}_i L(\theta_k)|$
    if $\hat{\theta}_k$ satisfies some specific condition, then STOP
    Set $\theta_{k+1} = \theta_k + \alpha_k(\hat{\theta}_k - \theta_k)$
  end for

2.2 Stochastic Variance Reduced Frank-Wolfe (SVRF)

Many variants have been proposed to improve the practical efficiency of SFW, most of these rely on modifying how the (unbiased) gradient estimator is obtained. Variance Reduced techniques originally tries to reduce the variance in gradient estimation of mini-batches in stochastic methodologies[2]. In this method, we will store a larger number of different set of parameters to model the variance reduction technique[2]. The main difference in comparison to Stochastic Frank-Wolfe is we change the gradient calculation formula(7) to:

$$\tilde{\nabla} F(\theta_k) \leftarrow \nabla F(\theta_0) + \frac{1}{b_k} \sum_{j=1}^{b_k} (\nabla f_{i_j}(\theta_k) - \nabla f_{i_j}(\theta_0)) \tag{7}$$

In above formula, each m (inner loop) step, we will store the full gradient of all samples and we use this one for correction of gradients of variable to improve the gradient estimation for stochastic methods. In addition, we reduce the gradient of the specific batch inside the inner loop based on snapshot that we took for variance reduction technique in this method. The full algorithm for SVRF can be found below.

Algorithm 3 Stochastic Variance Reduced Frank-Wolfe method for l_1-ball

Require: Starting from a point inside the region
  for t=0,...,S-1 **do**
    take snapshot $\theta_0 = \theta_t$ and compute $\nabla F(\theta_0)$
    for k=1,....m-1 **do**
      Uniformly sample i.i.d. $i_1.i_2, ..., i_b$ from $[1, .., n]$
      $\tilde{\nabla} F(\theta_k) \leftarrow \nabla F(x_0) + \frac{1}{b_k} \sum_{j=1}^{b_k} (\nabla f_{i_j}(x_k) - \nabla f_{i_j}(\theta_0))$
      Set $\hat{\theta}_k = diameter \times sign(-\tilde{\nabla}_{i_k} F(\theta_k))$, with $i_k = \arg\max_i |\tilde{\nabla}_i F(\theta_k)|$
      Set $\theta_{k+1} = \theta_k + \alpha_k(\hat{\theta}_k - \theta_k)$
    end for
  $\theta_{t+1} \leftarrow \theta_{K_t}$
  end for

3 Results

The results of the different variants of the stochastic Frank–Wolfe algorithm introduced above (SFW, SVRF) were run on 4 datasets. In this study, we use a fully connected nueral network with one hidden layer. In addition, we studied the two stochastic variant of frank-wolfe method with l_1 ball constraint and results for each iteration are reported in the figures. First of all we ran MNIST and Fashion MNIST datasets, which were used, for the purpose of rough comparison to experiments carried out by Sebastian Pokutta et al. [4]. However, in our case of this study, a binary classification was implemented, therefore we do not expect the results to match completely. Additionally, two newly outsourced datasets were used: a fruit dataset consisting of oranges and grapefruits species [1] and a toy dataset Moon that is made of half circle - moon clusters. Both of these were also used for binary classification.

For parameter initialization, a technique was used to ensure that the sum of the initialised variables is as close as possible to the half value of the chosen diameter. This ensures faster convergence as it is more likely that the minimum value is closer.

Datasets Hyperparameters The hyperparameters were chosen with the implementation of a grid search. In table 3 and 3 the hyperparameter result for both methods and for all the datasets are presented. Taking a look at the results presented in the mentioned tables below, it is presented how the different changes of the chosen parameters influence the loss. Although all the results are not presented here, it is important not to base the hyperparameter decision only on the final value of the cost, but also on the cost behaviour throughout all the iterations as well as the accuracy.

Table 2: SFW hyperparameter tuning, the activation function is relu

Data set	Learning rate	Batch size	l_1 ball diameter	Epochs	Hidden unit size
fashion mnist	0.001	128	5	20	32
moon	0.0008	32	3	10	16
fruit	0.001	32	20	10	64

In addition, for SVRF, we need to find best value for inne loop size. In the table below, there are some changes in hyperparameter with respect to SFW.

Table 3: SVRF hyperparameter tuning, the activation function is relu

Data set	Learning rate	Batch size	l_1 ball diameter	Epochs	Hidden unit size	inner loop size
fashion mnist	0.005	128	5	20	32	20
moon	0.003	32	3	10	16	10
fruit	0.001	32	20	10	64	10

Result Analysis The achieved accuracy and training loss is presented in the following tables. The chosen hyperparameters for all four mentioned datasets are presented in tables 4 and 5 respectively.

Table 4: Test set Accuracy comparisons of different Stochastic Frank-Wolfe variants

Dataset	SFW	SVRF
fashion mnist	94.9%	**95.2%**
moon	**87.3%**	84.5%
fruit	86.7%	**91.9%**

Table 5: Training Loss comparisons of different Stochastic Frank-Wolfe variants.

Dataset	SFW	SVRF
fashion mnist	0.613	**0.4371**
moon	0.625	**0.508**
fruit	0.415	**0.095**

Looking at the results in the tables above, it is clear that in the cases of all three datasets both methods give better results for SVRF rather than SFW in terms of the final training loss achieved as well as the accuracy, except in the case of the mood dataset. In the cases of both of the new datasets the results have similar tendencies. Looking at figures 2, 4 and 3, it is clear that SVRF converges faster in terms of the number of iterations, but takes more of the CPU time and a higher number of gradient calculations in comparison to SFW. The third plot from the top presents the sum of the variables, which has to stay within the given diameter. Looking at the behaviour of this sum in the Fruit dataset's SVRF instance, the behaviour shows that the minimum is potentially outside of the constraint, as the sum is increasing while reaching the minimum. Here we do not consider SFW since it's achieved cost converged to a higher value than the one when using SVRF method. This can be explained by the fact that there are multiple local minimas. However in the case of the Moon dataset, it can also be seen that the minimum is outside of the constrained region, as the sum for both of the methods stays close to the diameter length as the loss converges. Since the minimum is not inside the domain so using methods like frank-wolfe can be faster with respect to the gradient decent methods that need a projection in each step.

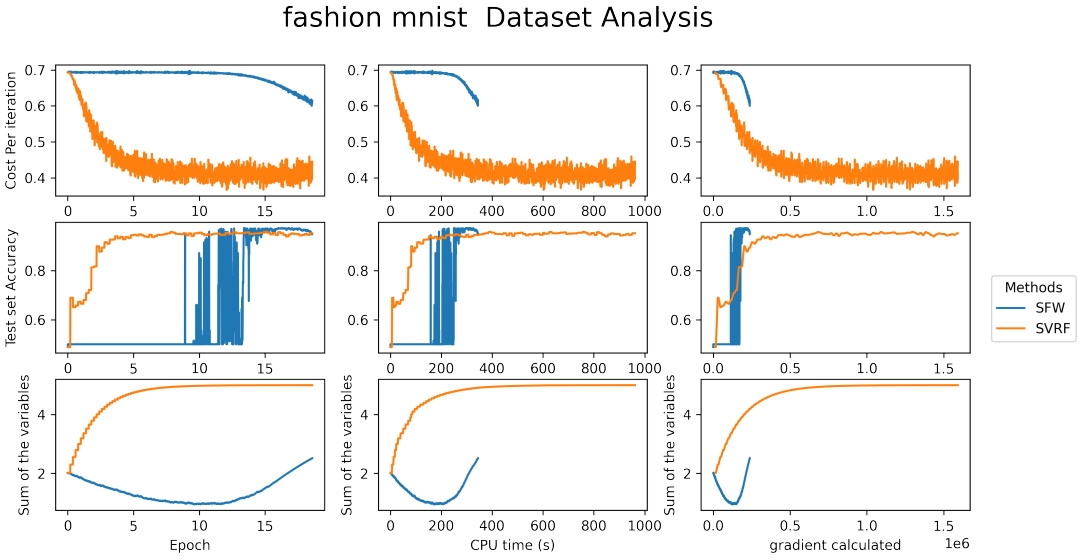

Figure 2: Comparing SFW and SVRF algorithms to train a fully connected Neural Network on fashion mnist dataset. The fully connected Neural Network consists of a single hidden layers of size 32 with ReLU activations. The l_1 -constraints with diameter 3. For SFW batch size was 128 and learning rate was 0.001 . For SVRF batch size was 128 and learning rate was 0.005 and initial point has been considered a point inside the convex domain. Results are reported over each run.

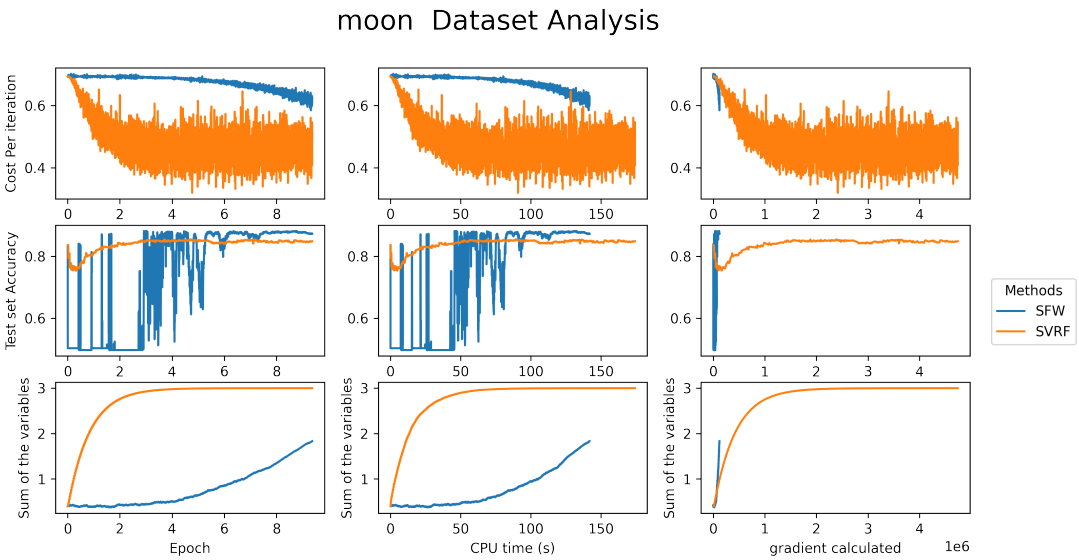

Figure 3: Comparing SFW and SVRF algorithms to train a fully connected Neural Network on the moon dataset. The fully connected Neural Network consists of a single hidden layers of size 16 with ReLU activations. The l_1 -constraints with diameter 3. For SFW batch size was 32 and learning rate was 0.0008 . For SVRF batch size was 32 and learning rate was 0.003 and initial point has been considered a point inside the convex domain. Results are reported over each run.

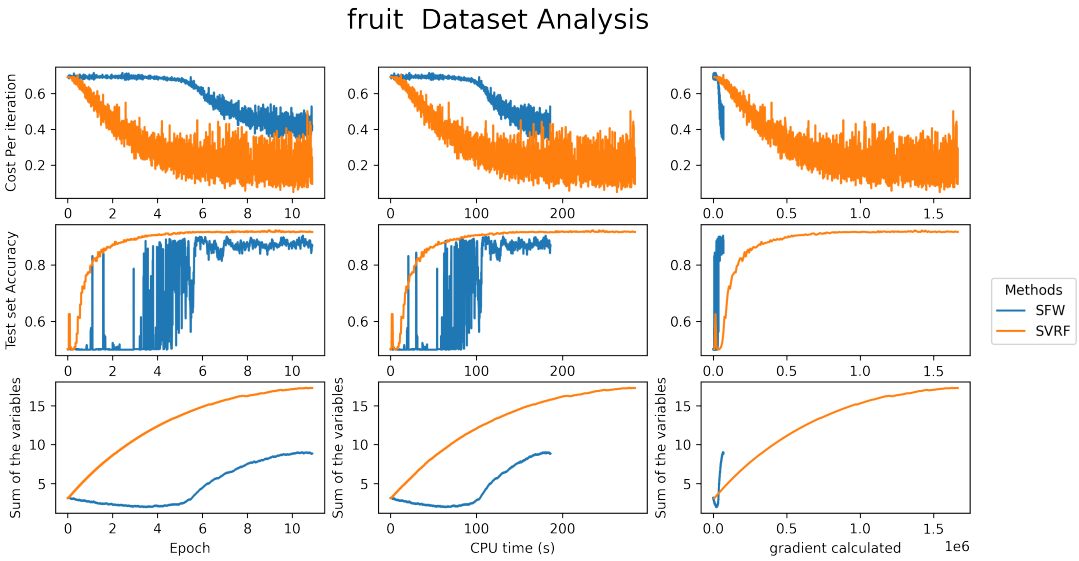

Figure 4: Comparing SFW and SVRF algorithms to train a fully connected Neural Network on fruit dataset. The fully connected Neural Network consists of a single hidden layers of size 64 with ReLU activations. The l_1 -constraints with diameter 20. For SFW batch size was 32 and learning rate was 0.001 . For SVRF batch size was 32 and learning rate was 0.001 and initial point has been considered a point inside the convex domain. Results are reported over each run.

4 Conclusion

The primary purpose of our work was studying the two stochastic variants of Frank-Wolfe method for a neural network. This was implemented on a one-layer fully connected neural network. The initial results that were taken

were averaged during the epoch and it did not show the noisy nature of the the stochastic methods. So later this was changed which resulted into having more noisy plots. Since only two methods are compared comparison, the noise does not create any problem for understanding the behavior of SFW and SVRF. However for this reason cost and accuracy has been plotted separately. Similarly as in the previously conducted studies, the results conclude that SVRF substantially reduces the number of iterations required for convergence in comparison to SFW. However, when it comes to the CPU time, SFW has the advantage.

References

[1] Fruit dataset from kaggle. `https://www.kaggle.com/datasets/joshmcadams/oranges-vs-grapefruit/code`. Accessed: 2022-09-14.

[2] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271. PMLR, 2016.

[3] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[4] Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Deep neural network training with frank-wolfe. 2020.

[5] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *2016 54th annual Allerton conference on communication, control, and computing (Allerton)*, pages 1244–1251. IEEE, 2016.

[6] Jiahao Xie, Zebang Shen, Chao Zhang, Boyu Wang, and Hui Qian. Efficient projection-free online methods with stochastic recursive gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6446–6453, 2020.