Non-convex Conditional Gradient Sliding

Chao Qu [1] **Yan Li** [2] **Huan Xu** [2]

Abstract

We investigate a projection free optimization method, namely non-convex conditional gradient sliding (NCGS) for non-convex optimization problems on the batch, stochastic and finite-sum settings. Conditional gradient sliding (CGS) method, by integrating Nesterov's accelerated gradient method with Frank-Wolfe (FW) method in a smart way, outperforms FW for convex optimization, by reducing the amount of gradient computations. However, the study of CGS in the non-convex setting is limited. In this paper, we propose the non-convex conditional gradient sliding (NCGS) methods and analyze their convergence properties. We also leverage the idea of variance reduction from the recent progress in convex optimization to obtain a new algorithm termed *variance reduced NCGS* (NCGS-VR), and obtain faster convergence rate than the batch NCGS in the finite-sum setting. We show that NCGS algorithms outperform their Frank-Wolfe counterparts both in theory and in practice, for all three settings, namely the batch, stochastic and finite-sum setting. This significantly improves our understanding of optimizing non-convex functions with complicated feasible sets (where projection is prohibitively expensive).

1. Introduction

This paper studies *non-convex* optimization problems with *complicated feasible sets*. Specifically, we consider the following problem

$$\min_{\theta\in\Omega} F(\theta), \tag{1}$$

where the objective function $F(\theta)$ is non-convex and L smooth, and Ω is a convex compact set.

[*]Equal contribution [1]EE faculty, Technion, Israel [2]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, USA. Correspondence to: Chao Qu <chaoqu.technion@gmail.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

Besides this general form, we also consider a stochastic setting and a finite-sum setting. In the stochastic setting, we assume $F(\theta)=E_\xi f(\theta,\xi)$, where $f(\theta,\xi)$ is smooth and non-convex. In the finite-sum case, we study the following problem

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

where each $f_i(\theta)$ is non-convex and L smooth, Ω is a convex compact set. Here, we are interested in the case where n, i.e., the number of training examples, is very large.

We focus on the case where the feasible set Ω is complicated, in the sense that projection onto Ω is expensive (for instance, the projection on the trace norm ball), or even computationally intractable (Collins et al., 2008). To alleviate such difficulty, the Frank-Wolfe method (Frank & Wolfe, 1956) (a.k.a. conditional gradient method), which was initially developed for the convex problem in 1950s, has attracted much attention again in machine learning community recently, due to its projection free property (Jaggi, 2013). In each iteration, the Frank-Wolfe algorithm calls a first-order oracle to get $\nabla F(\theta)$, and then calls a linear oracle in the form $\arg\min_{\theta\in\Omega}\langle\theta, g\rangle$, which avoids the projection operation.

This setup is motivated by several popular problems in machine learning, wherein the above linear optimization is easy but the projection is much more computationally demanding. The example *par excellence* is the nuclear norm constraint which is widely used in multi-task learning, mutli-class classification, recommendation systems and matrix learning. We briefly explain some of them: Suppose there are m tasks and each column i of a matrix Θ represents a task i, one popular multi-task learning formulation proposed by Pong et al. (2010) has the following form.

$$\begin{aligned}\min_{\Theta,b}\quad & \sum_{i=1}^{m}\sum_{j=1}^{n_i}\ell(y_j^i,\theta_i^T x_j^i+b_i)\\ \text{subject to}\quad & \|\Theta\|_*\le R,\end{aligned} \tag{2}$$

where $\ell(\cdot,\cdot)$ is the loss function which can be potentially non-convex, n_i is the number of samples in each task, $\Theta=[\theta_1,...,\theta_m]$, and $\|\cdot\|_*$ is the nuclear norm constraint to encourage the low rank property. In the multiclass classification problem, suppose there are n training examples

$(x_i, y_i)_{i=1,...,n}$, where x_i is a feature vector and y_i is the label. The goal is to find an accurate linear predictor with parameter $\Theta = [\theta_1, ..., \theta_h]$. In Zhang et al. (2012) and Dudik et al. (2012), multivariate logistic loss with nuclear norm regularization is proposed with the following form

$$f_i(\Theta) = \log(1 + \sum_{\ell \neq y_i} \exp(\theta_\ell^T x_i - \theta_{y_i}^T x_i)),$$

and $\Omega = \|\Theta\|_* \leq R$. The logistic loss can be replace by a non-convex loss function due to the superior robustness and classification accuracy of the non-convex loss (Mei et al., 2016).

Apart from the nuclear norm constraint, other examples of complicated feasible sets include polytopes with exponentially many facets, often resulted from combinatorial problems (e.g., the convex hull of all spanning trees). We refer reader to Garber & Hazan (2013) and Lacoste-Julien & Jaggi (2015) for details.

In the convex case, it is well known that for the Frank-Wolfe method to achieve ϵ-solution, $\mathcal{O}\big(\frac{1}{\epsilon}\big)$ iterations are required, if $F(\theta)$ is convex and smooth. This rate is significantly worse than the optimal rate $\mathcal{O}\big(1/\sqrt{\epsilon}\big)$ for smooth convex problems (Nesterov, 2013), which raises a question whether this complexity bound $\mathcal{O}(\frac{1}{\epsilon})$ is improvable. Unfortunately, the answer is no in the general setting (Lan, 2013; Guzmán & Nemirovski, 2015) and improved results can only be obtained under stronger assumptions, see e.g., Garber & Hazan (2013; 2015); Lacoste-Julien & Jaggi (2015). Lan & Zhou (2016) proposed the conditional gradient sliding method (CGS) which combines the idea of Nesterov's accelerated gradient with the Frank-Wolfe method. While the number of linear oracle calls remains same, the number of gradient computations (the first order oracle) is significantly improved from $\mathcal{O}(\frac{1}{\epsilon})$ to $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$. Under the strongly convex assumption, this can be further improved to $\mathcal{O}(\log(1/\epsilon))$ by using the restarting techniques (Lan & Zhou, 2016).

Recently, non-convex optimization has attracted lots of attentions and becomes the frontier of the machine learning, where a partial list of applications includes robust regression and classification (Mei et al., 2016), dictionary learning (Mairal et al., 2009), phase retrial (Candes et al., 2015) and training the neural network (Goodfellow et al., 2016). Therefore, the convergence on non-convex Frank-Wolfe method has been studied, under the batch, stochastic and finite-sum setting (Lacoste-Julien, 2016; Reddi et al., 2016b). A natural question arises: can we use similar technique of convex conditional gradient sliding in the non-convex conditional gradient sliding and improve the complexity on the first order oracle? This paper provides an affirmative answer.

Summary of contributions: We propose the non-convex conditional gradient sliding (NCGS) method and provides a convergence analysis in the batch and the stochastic setting. Compared to the convex CGS, the difficulty comes from the analysis of non-convexity. In the finite-sum setting, we propose the variance reduction non-convex gradient sliding method (NCGS-VR) which achieves faster convergence than the batch one. We need carefully balance the number of call on the linear oracle and first order oracle. All result are summarized in Table 1,2, 3 (with red color). Table 1 compares the result of non-convex conditional gradient sliding with the non-convex Frank-Wolfe method in the batch setting. Table 2 compares our method with SAGAFW and SVFW (Reddi et al., 2016b) in the stochastic setting. In Table 3, our method leverages the popular stochastic variance reduction technique. We compare it with the stochastic variance reduction Frank-Wolfe method in (Reddi et al., 2016b). To the best of our knowledge, our algorithms outperform Frank-Wolfe method in all corresponding setting. We defer the detailed comparison to the related work subsection. Please see Section 2 for the formal definition of first order oracle (FO), stochastic first order oracle (SFO), Incremental First Order Oracle (IFO) and linear oracle (LO).

We remark that the convergence criterion used in paper is different from that in Frank-Wolfe. In our paper, we follow the criterion $\|\nabla F(\theta)\|^2 \leq \epsilon$ (See section 2.3 for more precise definition on convergence criteria), as that in most non-convex optimization work (Lan, 2013; Allen-Zhu & Hazan, 2016; Reddi et al., 2016c; Nesterov, 2013), while the Frank-Wolfe method uses the Frank-Wolfe gap. Understanding the precise relationship between these convergence criteria is an important direction for future research.

Algorithm	FO complexity	LO complexity
NCGS	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$
FW	$\mathcal{O}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon^2)$

Table 1. Comparison of complexity of algorithms in the batch setting.

Algorithm	SFO complexity	LO complexity
NCGS	$\mathcal{O}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon^2)$
SAGAFW	$\mathcal{O}(1/\epsilon^{\frac{8}{3}})$	$\mathcal{O}(1/\epsilon^2)$
SVFW	$\mathcal{O}(1/\epsilon^{\frac{10}{3}})$	$\mathcal{O}(1/\epsilon^2)$

Table 2. Comparison of complexity of algorithms in the stochastic setting.

Related work

The classical Frank-Wolfe method considers optimizing a smooth convex function $F(\theta)$ over a polyhedral set and enjoys $\mathcal{O}(1/\epsilon)$ convergence rate (Frank & Wolfe, 1956; Jaggi, 2013). Recent work (Garber & Hazan, 2013; 2015) proves faster convergence rate under additional assumptions. Conditional gradient sliding was proposed in (Lan & Zhou, 2016), aiming at minimizing a convex objective function.

Algorithm	IFO complexity	LO complexity
NCGS	$\mathcal{O}(n/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$
FW	$\mathcal{O}(n/\epsilon^2)$	$\mathcal{O}(1/\epsilon^2)$
NCGS-VR	$\mathcal{O}(n+\frac{n^{\frac{2}{3}}}{\epsilon})$	$\mathcal{O}(1/\epsilon^2)$
SVFW	$\mathcal{O}(n+\frac{n^{\frac{2}{3}}}{\epsilon^2})$	$\mathcal{O}(1/\epsilon^2)$

Table 3. Comparison of complexity of algorithms in the finite-sum setting. Since we need to evaluate n gradients each iteration in NCGS and FW, the IFO complexity of NCGS and FW are n× results in table 1.

While our high level algorithmic idea is the same, the analysis differs significantly due to the non-convexity of the objective function.

Most existing works on analyzing non-convex optimization solve the problem with the projection or the proximal operation. Hence we list some representative works below. Ghadimi & Lan (2013) investigate SGD in the non-convex setting. They extend Nesterov's acceleration method in the constrained stochastic optimization. The performance on non-convex stochastic variance reduction method is analyzed in Reddi et al. (2016a); Allen-Zhu & Hazan (2016); Shalev-Shwartz (2016); Allen-Zhu & Yuan (2016). Note that the stochastic variance reduction techniques are first introduced for solving convex optimization problems (Xiao & Zhang, 2014; Johnson & Zhang, 2013; Xiao & Zhang, 2014).

There are very few work on projection free methods for non-convex optimization. The early work in Bertsekas (1999) proves the asymptotic convergence of the Frank-Wolfe method to a stationary point, but the convergence rate is not studied. Lacoste-Julien (2016) establishes the convergence rate of $\mathcal{O}(1/\epsilon^2)$ for the Frank-Wolfe method in the (batch) non-convex setting, under the criteria of Frank-Wolfe gap. In his work, both FO and LO complexity are shown to be $\mathcal{O}(1/\epsilon^2)$. In contrast, for our proposed NCGS, the FO complexity is $\mathcal{O}(1/\epsilon)$ and the LO complexity is $\mathcal{O}(1/\epsilon^2)$. Recent work on Frank-Wolfe method for non-convex, stochastic setup shows that the SFO complexity and the LO complexity are $\mathcal{O}(1/\epsilon^{\frac{10}{3}})$ and $\mathcal{O}(1/\epsilon^2)$ respectively for SVFW, and $\mathcal{O}(1/\epsilon^{\frac{8}{3}})$ and $\mathcal{O}(1/\epsilon^2)$ for SAGAFW (Reddi et al., 2016b). Our SFO and LO on the same setting are $\mathcal{O}(1/\epsilon^2)$ and $\mathcal{O}(1/\epsilon^2)$ respectively. In the finite sum setting, our variance reduction NCGS(NCGS-VR) has IFO complexity $\mathcal{O}(n+\frac{n^{\frac{2}{3}}}{\epsilon})$, while the state of the art variance reduced FW has IFO complexity $\mathcal{O}(n+\frac{n^{\frac{2}{3}}}{\epsilon^2})$ and the same LO complexity (Reddi et al., 2016b). Thus, it is clear that for all three settings, we improved upon the best known results in literature in terms of reducing computation for gradient evaluation.

2. Preliminary

2.1. Oracle model

We consider the following set of Oracles:

- First Order Oracle (FO): given a θ, the FO returns $\nabla_\theta F(\theta)$.
- Stochastic First Order Oracle (SFO): For a function $F(\theta) = \mathbb{E}_\xi f(\theta,\xi)$ where $\xi \sim P$, a SFO returns the stochastic gradient $G(\theta_k,\xi_k) = \nabla_\theta f(\theta_k,\xi_k)$ where ξ_k is a sample drawn i.i.d. from P in the k-th call.
- Incremental First Order Oracle (IFO): For the setting $F(\theta) = \frac{1}{n}\sum_{i=1}^n f_i(\theta)$, an IFO samples $i \in [n]$ and returns $\nabla_\theta f_i(\theta)$.
- Linear oracle (LO): the LO solves the following problem $\arg\min_{\theta\in\Omega}\langle\theta, g\rangle$ for a given vector g.

Thought out the paper, the complexity of FO, SFO, IFO, LO denotes the number of call of them to obtain a solution with "ϵ accuracy" (see Section 2.3 for details).

2.2. Assumptions

$F(\theta)$ is L smooth, if $\|\nabla F(\theta_1) - \nabla F(\theta_2)\| \le L\|\theta_1 - \theta_2\|$. This definition is equivalent to the following form:

$$-\frac{L}{2}\|\theta_1-\theta_2\|^2 \le F(\theta_1) - F(\theta_2) - \langle \nabla F(\theta_2), \theta_1-\theta_2\rangle \le \frac{L}{2}\|\theta_1-\theta_2\|^2, \quad \forall \theta_1,\theta_2 \in \Omega.$$

We say $F(\theta)$ is ℓ lower smooth if it satisfies

$$-\frac{l}{2}\|\theta_1-\theta_2\|^2 \le F(\theta_1) - F(\theta_2) - \langle \nabla F(\theta_2), \theta_1-\theta_2\rangle, \quad \forall \theta_1,\theta_2 \in \Omega.$$

Intuitively, the lower smoothness quantified the amount of "non-convexity" of the function. Clearly, the L smooth assumption trivially implies l lower smoothness for $l = L$. However, in some cases, the non-convexity l is much smaller than L and we will show how it affects some results in our theorem.

We then define prox-mapping type functions $\psi(x,\omega,\gamma)$:

$$\psi(x,\omega,\gamma) = \arg\min_{\theta\in\Omega}\langle\omega,\theta\rangle + \frac{1}{2\gamma}\|\theta - x\|^2.$$

It is closely related to the projected gradient by setting $w = \nabla F(\theta)$, γ by the stepsize and $x = \theta_k$. We assume $\|\psi(x,\omega,\gamma)\| \le M$ for all $\gamma \in (0,\infty)$ and $x \in \Omega$ and $\omega \in R^p$ following that in (Lan, 2013). Since in our work Ω is compact and convex, this assumption is satisfied.

For the stochastic setting, we make the following additional assumptions: For any $\theta \in \mathbb{R}^p$ and $k > 1$, we have

$$\begin{aligned}
&(1). \qquad \mathbb{E}G(\theta, \xi_k) = \nabla F(\theta)\\
&(2). \qquad \mathbb{E}\|G(\theta, \xi_k) - \nabla F(\theta)\|^2 \le \sigma^2,
\end{aligned}$$

i.e., the stochastic gradient $G(\theta, \xi_k)$ is unbiased and has bounded variance.

In the finite-sum setting, we assume each $f_i(\theta)$ is L smooth.

2.3. Convergence criteria

Conventionally, the convergence criterion in non-convex optimization defines a *solution with ϵ accuracy* as $\|\nabla F(\theta)\|^2 \le \epsilon$ (Lan & Zhou, 2016; Nesterov, 2013). However when the problem has constraints, it needs a different termination criterion based on the gradient mapping (Lan & Zhou, 2016). This is a natural extension of gradient, since if there is no constraint, it reduces to the gradient. The gradient mapping is defined as follows

$$g(\theta, \nabla F(\theta), \gamma) = \frac{1}{\gamma}(\theta - \psi(\theta, \nabla F(\theta), \gamma)).$$

Through out the paper, we use $g(\theta, \nabla F(\theta), \gamma)$ as the convergence criterion, i.e., we want to find the solution θ such that $\|g(\theta, \nabla F(\theta), \gamma)\|^2 \le \epsilon$.

Notice there is another criterion, called Frank-Wolfe gap $\max_{x\in\Omega}\langle x - \theta_k, -\nabla F(\theta_k)\rangle$, in some recent analysis of non-convex Frank-Wolfe methods (Lacoste-Julien, 2016; Reddi et al., 2016a), which was initially used in the convex Frank-Wolfe method. However, in this paper, we follow the definition on gradient mapping, since it is a natural generalization of gradient.

3. Batched non-convex conditional gradient sliding

3.1. Algorithm

Before we present the algorithm of non-convex conditional gradient sliding, we present a procedure *condg* in Algorithm 1, which will be used as a subroutine in all three (i.e., batch, stochastic and finite-sum) settings.

Algorithm 2 presents the non-convex conditional gradient sliding in the batch setting. Notice it needs to call the procedure $condg$. There are two options to update θ^{ag}, and we provide the theoretical guarantees for both of them in Theorem 1.

3.2. Theoretical result

Theorem 1. *Suppose the objective function $F(\theta)$ satisfies the assumption in section 2, where L is the smoothness parameter and l is the lower smoothness parameter, then if we set $\alpha_k = \frac{2}{k+1}$, $\beta_k = \frac{1}{2L}$, $\lambda_k = \beta_k$, $\eta_k = \frac{1}{N}$ in option I of Algorithm 2, we have*

$$\begin{aligned}
&\min_{k=1,\dots,N} \|g(\theta_{k-1}, \nabla F(\theta_k^{md}), \lambda_k)\|^2\\
&\le \frac{12L(F(\theta_0) - F(\theta^*)) + 16L}{N},
\end{aligned} \tag{3}$$

where θ^* is the optimal solution of equation 1.

In option II, we set $\alpha_k = \frac{2}{k+1}$, $\beta_k = \frac{1}{2L}$, $\lambda_k = k\beta_k/2$, $\eta_k = \frac{1}{N}$, $\chi_k = \frac{1}{N}$, M is the positive constant defined in our section 2.2, then we have

$$\begin{aligned}
&\min_{k=1,\dots,N} \|g(\theta_{k-1}, \nabla F(\theta_k^{md}), \beta_k)\|^2\\
&\le \frac{192L^2\|\theta_0 - \theta^*\|^2}{N^2(N+1)} + \frac{48lL}{N}(\|\theta^*\|^2 + 2M^2) + \frac{96L}{N}.
\end{aligned} \tag{4}$$

Some remarks are in order:

- Notice in the procedure of $condg$, we solve the sub problem with accuracy η. The choice of η is important: If η is chosen too small, we need too many calls on LO. On the other hand, if η is too large, the algorithm may not converge.

Algorithm 1 Procedure of $u^+ = condg(l, u, \gamma, \eta)$

```
1.$u_1 = u$ and $t = 1$.
2.$v_t$ be an optimal solution for the subproblem
    $V(u_t) = \max_{x\in\Omega}\langle l + \frac{1}{\gamma}(u_t - u), u_t - x\rangle$
3.if $V(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure
4.$u_{t+1} = (1 - \xi_t)u_t + \xi_t v_t$ with $\xi_t = \min\{1, \frac{\langle \frac{1}{\gamma}(u - u_t) - l, v_t - u_t\rangle}{\frac{1}{\gamma}\|v_t - u_t\|^2}\}$.
Set $t \leftarrow t + 1$ and go to step 2.
end procedure
```

Algorithm 2 Non-convex conditional gradient sliding (NCGS)

```
Input: Step size $\alpha_k$, $\lambda_k$, $\beta_k$, smoothness parameter $L$.
Initialization: $\theta_0^{ag} = \theta_0$, k=1.
for $k = 1, ..., N$ do
  update: $\theta_k^{md} = (1 - \alpha_k)\theta_{k-1}^{ag} + \alpha_k\theta_{k-1}$
  update: $\theta_k = condg(\nabla F(\theta_k^{md}), \theta_{k-1}, \lambda_k, \eta_k)$
  update:
  option I: $\theta_k^{ag} = \theta_k^{md} - \beta_k\tilde{g}(\theta_{k-1}, \nabla F(\theta_k^{md}), \lambda_k, \eta_k)$,
  where $\tilde{g}(\theta_{k-1}, \nabla F(\theta_k^{md}), \lambda_k, \eta_k) := \frac{\theta_{k-1} - \theta_k}{\lambda_k}$.
  option II: $\theta_k^{ag} = condg(\nabla F(\theta_k^{md}), \theta_k^{md}, \beta_k, \chi_k)$.
end for
```

- The FO complexities of option I and II are order wise equivalent, namely, $\mathcal{O}(1/\epsilon)$.
- We now examine each terms of the convergence guarantee of Option II: the term $\frac{L^2\|\theta_0-\theta^*\|^2}{N^2(N+1)}$ corresponds to the convex part of the function. The term $\frac{Ll}{N}(\|\theta^*\|^2+2M^2)$ corresponds to the non-convex part of the function. And the last term L/N corresponds to the procedure of condg.
- When the objective function a has finite-sum form with n term, the IFO complexity of NCGS is $\mathcal{O}(\frac{n}{\epsilon})$. We will improve this complexity using stochastic variance reduction techniques in Section 5.

Theorem 1 presents the convergence in terms of iteration number, which we transfer to the FO and LO complexity in the following corollary.

Corollary 1. *Under the same condition of theorem 1. In option I and II of algorithm 2, to achieve the accuracy ϵ, the FO complexity is $\mathcal{O}(1/\epsilon)$ and the LO complexity is $\mathcal{O}(\frac{1}{\epsilon^2})$.*

- Our algorithm has the same LO complexity with FW but improves the FO complexity from $\mathcal{O}(\frac{1}{\epsilon^2})$ to $\mathcal{O}(\frac{1}{\epsilon})$.

4. Stochastic non-convex conditional gradient sliding

In this section we consider the following stochastic optimization problem:

$$\min_{\theta\in\Omega} F(\theta) := E_\xi f(\theta,\xi). \tag{5}$$

4.1. Algorithm

A natural way to extend batch NCGS method to the stochastic case is to replace the exact gradient $\nabla F(\theta)$ in Algorithm 2 by the stochastic gradient $G(\theta,\xi)$. However it is shown in Ghadimi & Lan (2016) that mini-batch stochastic gradient is necessary to guarantee the convergence. We incorporate this technique in the stochastic NCGS . In particular, we define $\bar{G}_k = \frac{1}{m_k}\sum_{i=1}^{m_k} G(x_k^{md},\xi_{k,i})$.

Notice, by our assumption in Section 2, we have

$$\mathbb{E}\bar{G}_k = \frac{1}{m_k}\sum_{i=1}^{m_k}\mathbb{E}G(\theta_k^{md},\xi_{k,i}) = \nabla F(\theta_k^{md}),$$

and

$$\mathbb{E}\|\bar{G}_k - \nabla F(\theta_k^{md})\|^2 \le \frac{\sigma^2}{m_k}. \tag{6}$$

We present the stochastic NCGS in Algorithm 3. Notice we have a randomized termination criterion on the total iteration R.

Algorithm 3 Stochastic Non-convex conditional gradient sliding

```
**Input:** Step size $\alpha_k, \lambda_k, \beta_k$, smoothness parameter $L$, a probability mass function $P_R(\cdot)$ with $Prob\{R=k\} = p_k, k=1,...,N$.
Initialization: $\theta_0^{ag} = \theta_0$, k=1.
Let $R$ be a random variable chosen according to $P_R(\cdot)$.
for $k = 1,...,R$ do
  update: $\theta_k^{md} = (1-\alpha_k)\theta_{k-1}^{ag} + \alpha_k\theta_{k-1}$.
  update: $\theta_k = condg(\bar{G}_k, \theta_{k-1}, \lambda_k, \eta_k)$.
  update: $\theta_k^{ag} = condg(\bar{G}_k, \theta_k^{md}, \beta_k, \chi_k)$.
end for
```

4.2. Theoretical Result

In the following theorem, we carefully choose the value of step size and the tolerance in the procedure $condg$ to guarantee the convergence of the algorithm.

Theorem 2. *Suppose $F(\theta)$ is L smooth and l lower smooth, σ^2 is the variance defined in Section 2. In Algorithm 3, set $\alpha_k = \frac{2}{k+1}$, $\beta_k = \frac{1}{2L}$, $\lambda_k = \frac{k\beta_k}{2}$, $\eta_k = \chi_k = \frac{1}{N}$ and $m_k = k$, and set $p_k = \frac{\Gamma_k^{-1}}{\sum_{k=1}^N \Gamma_k^{-1}}$, where $\Gamma_k = \frac{2}{k(k+1)}$, then we have*

$$\mathbb{E}[\|g(\theta_R^{md},\bar{G}_R,\beta_R)\|^2] \le 192L\Big(\frac{4L\|\theta_0-\theta^*\|^2}{N^2(N+1)} + \frac{l}{N}(\|\theta^*\|^2+2M^2) + \frac{1}{N} + \frac{3\sigma^2}{2LN}\Big), \tag{7}$$

where θ^* is the optimal solution of equation 5.

Remarks:

- Compare this result with its batch counterpart, namely, Theorem 1, we see there is a additional term $\frac{c\sigma^2}{N}$ due to the variance of the gradient.
- The mini-batch size is set as $m_k = k$, i.e., increasing with the iteration of the algorithm. This is chosen to guarantee the convergence with a fast rate.

Theorem 2 implies the following corollary of LO and SFO complexity.

Corollary 2. *Under the same setting as Theorem 2, SFO and LO complexities in algorithm 3 are $\mathcal{O}(1/\epsilon^2)$ and $\mathcal{O}(1/\epsilon^2)$ respectively.*

- Notice in algorithm 3, we use mini-batches to calculate $\bar{G}_k$.Thus even the number of iteration of the stochastic non-convex conditional gradient sliding is the same as the batch one, it needs more calls of SFO.

- To the best of our knowledge, the corresponding Frank-Wolfe method in Reddi et al. (2016b) has SFO complexity $\mathcal{O}(1/\epsilon^{\frac{8}{3}})$. Our algorithm has the same LO complexity while improves the SFO complexity.

5. Variance reduction nonconvex conditional gradient sliding in finite-sum setting

The stochastic variance reduction technique has been very successful in optimizing convex functions in the form of finite sums. In this section we incorporate it with our non-convex conditional gradient sliding and propose NCGS-VR in Algorithm 4.

5.1. Algorithm

We consider minimizing a finite sum problem as the following

$$\min_{x\in\Omega} F(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta), \tag{8}$$

where each f_i is possibly non-convex, and smooth with parameter L. If we view the finite-sum problem as a special case of batch problem, then use Algorithm 2, we have IFO complexity $\mathcal{O}(\frac{n}{\epsilon})$. Variance reduction technique has been proposed for finite sum problem to reduce the dependence of IFO complexity on number of components n. We incorporate this technique into NCGS. At the out looper, we calculate the full gradient and use it in the inner loop to reduce the variance of the stochastic gradient. Then we call the procedure *condg*. As we show below our new algorithm (Algorithm 4) achieves IFO complexity $\mathcal{O}(n + \frac{n^{\frac{2}{3}}}{\epsilon})$. To the best of our knowledge, this outperforms Frank-Wolfe type algorithms for the non-convex finite-sum problem. Notice in Algorithm 4, different from Algorithm 2 and 3, we do not apply Nestrerov's acceleration step. Whether the acceleration step can further improve the rate of convergence (e.g., exploit the lower smoothness) in this setting is left for future research.

5.2. Theoretical result

Theorem 3. *Suppose $f_i(\theta)$ is non-convex and L smooth, set $b = n^{\frac{2}{3}}$ in Algorithm 4, $\lambda_t = \frac{1}{3L}$, $m = n^{\frac{1}{3}}$ and T is a multiple of m, $\eta = \frac{1}{T}$. Then for output θ_a we have:*

$$\mathbb{E}[\|g(\theta_\alpha, \nabla F(\theta_\alpha), \lambda)\|^2] \leqslant \frac{18L(f(\theta_0) - f(\theta^\star) + 1)}{T}$$

where $\theta^\star$ is an optimal solution to (8).

- If $b = 1$ and $m = n$, then v_t^s reduces to the regular stochastic variance reduced gradient. However this means in every step we sample one data point and then call *condg*, which may deteriorate the performance of the algorithm.

Algorithm 4 Variance reduction Non-convex conditional gradient sliding (NCGS-VR)

```
Input: $\tilde{\theta}_0 = \theta_m^0 = \theta_0 \in \mathbb{R}^d$, epoch length $m$, stepsize $\lambda_t$,
tolerance $\eta$, minibatch size $b$, iteration limit $T$, $S = \frac{T}{m}$.
for $s = 0, ..., S-1$ do
  $\theta_0^{s+1} = \theta_m^s$.
  $g^{s+1} = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{\theta}^s)$.
  for $t = 0, \ldots, m-1$ do
    Pick $I_t$ uniformly from $\{1, \ldots, n\}$ with replacement
    such that $|I_t| = b$.
    $v_t^{s+1} = \frac{1}{b}\sum_{i\in I_t}(\nabla f_{i_t}(\theta_t^{s+1}) - \nabla f_{i_t}(\tilde{\theta}^s)) + g^{s+1}$.
    $\theta_{t+1}^{s+1} = condg(v_t^{s+1}, \theta_t^{s+1}, \lambda_t, \eta)$.
  end for
  $\tilde{\theta}^{s+1} = \theta_m^{s+1}$
end for
Output: $\theta_\alpha$ is chosen uniformly at random from
$\{\{\theta_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$.
```

- The minibatch gradient with size $b = n^{2/3}$ and iteration length $m = n^{1/3}$ are carefully chosen to guarantee the convergence of the algorithm and fast rate.

Theorem 3 leads to the following results on IFO and LO complexity.

Corollary 3. *Set the parameters set as in Theorem 3, the IFO and LO complexities of Algorithm 4 are $\mathcal{O}(n+\frac{n^{\frac{2}{3}}}{\epsilon})$ and $\mathcal{O}(\frac{1}{\epsilon^2})$ respectively to achieve $\mathbb{E}[\|g(\theta_\alpha, \nabla F(\theta_\alpha), \lambda)\|^2] \leqslant \epsilon$.*

- The stochastic variance reduction Frank-Wolfe method has the IFO complexity $\mathcal{O}(n+\frac{n^{2/3}}{\epsilon^2})$, while our method has the IFO complexity $\mathcal{O}(n + \frac{n^{2/3}}{\epsilon})$. The LO complexity for both algorithms are the same.

6. Simulation Result

In this section we test our algorithm in the batch (NCGS) and finite-sum setting (NCGS-VR) and compare them with Frank-Wolfe counterparts (FW and SVFW (Reddi et al., 2016b)), as well as SVRG, a projection based algorithm.

6.1. Synthetic dataset

In this section, we first use a toy example on matrix completion to examine the convergence of the *gradient mapping*, which is the convergence criteria for the algorithm. Notice that in practice, the objective function value is typically a more relevant metric, and hence we report such results using a multitask learning problem.

6.1.1. Matrix completion

We consider a toy matrix completion problem for our simulation and observe the magnitude of gradient mapping. In particular, we optimize the following trace norm constrained non-convex problem using the candidate algorithms.

$$\min_{\theta} \sum_{(i,j)\in\Omega} f_{i,j}(\theta) \quad \text{s.t.} \quad \|\theta\|_* \le R, \tag{9}$$

where Ω is the set of observed entries, $f_{i,j} = \big(1 - \exp(-\frac{(\theta_{i,j}-Y_{i,j})^2}{\sigma})\big)$, $Y_{i,j}$ is the observation of (i,j)'s entry, $\|\cdot\|_*$ is the nuclear norm. Here $f_{i,j}$ is a smoothed ℓ_0 loss with enhanced robustness to outliers in the data, thus it can solve sparse+low rank matrix completion in Chandrasekaran et al. (2009). Obviously, $f_{i,j}$ is non-convex and satisfies assumptions in our algorithm 2,3,4. We compare our non-convex conditional gradient sliding method with the Frank-Wolfe method in Fig 1. Particularly, we report the result of the batch setting in the left panel. The dimension of the matrix is 200×200, rank $r = 5$, the probability of observing each entry is 0.1. The sparse noise is sampled uniformly from $[-3, 3]$. Each entry is corrupted by noise with probability 0.05. We set $\sigma = 1$, $R = 5$ in Problem (9). We observe that our algorithm 2 (NCGS) clearly outperforms the non-convex Frank-Wolfe method (FW). In the right panel, we treat Problem (9) as a finite-sum problem, and thus solve it using Algorithm 4 (NCGS-VR) and compare the performance with the SVFW (Reddi et al., 2016b). We set the dimension of the matrix as 400×400, $rank\ r = 8$, $\sigma = 1$, $R = 8$. The way to generate sparse noise and the probability to observe the entry are same with the setting of batch case. We observe that our NCGS-VR uses around 50 cpu-time to achieve 10^{-3} accuracy of squared gradient mapping, while SVFW needs more than 300 cpu-time.

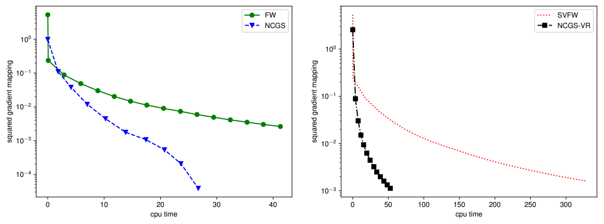

Figure 1. Left figure: NCGS and non-convex Frank-Wolfe method. Right figure: non-convex stochastic variance reduction Frank-Wolfe and NCGS-VR. The x-axis is the cpu-time with unit second, y-axis is the squared gradient mapping.

Notice in this example, NCGS-VR is not necessary faster than NCGS, since computing the gradient is very cheap.

6.1.2. Non-convex multitask learning

In this section, we consider the non-convex multitask learning problem and compare the performance of NCGS, NCGS-VR, non-convex Frank-Wolfe (FW) (Lacoste-Julien, 2016), stochastic variance reduction Frank-wolfe (SVFW) (Reddi et al., 2016b) and SVRG (Johnson & Zhang, 2013). In SVRG, we update θ and then project it back to the feasible set of the nuclear norm constraint. In the experiment we apply the mini-batch technique on SVRG. The reason is that in regular SVRG (with mini-batch size =1) it samples one data point and then calls the the procedure $condg$, resulting in very slow convergence. We choose the mini-batch $b = n^{2/3}$ and $m = n^{1/3}$ as that in Reddi et al. (2016a).

We first generate m different covariance matrices Σ_i, $i = 1, ..., m$, according to $\Sigma_i = U_i D_i U_i^T$, where $D_i \in \mathbb{R}^{d\times d}$ is a diagonal matrix with each entry drawn uniformly from $(1, 2)$, and $U_i \in \mathbb{R}^{d\times d}$ is a random matrix with each entry drawn from $N(0, 1)$. The feature vectors of task (i, k) are generated from the Normal distribution $\mathcal{N}(0, \Sigma_i + \Delta\Sigma_{ik})$, $k = 1, ...K$, where $\Delta\Sigma_{ik} = U_i \Delta D_i U_i^T$, ΔD_i is a small perturbation of D_i uniformly sampled from $(0, 0.1)$. Thus for each i, there are K similar tasks. Totally we have $m \times K$ tasks, where some of them are similar and others may be different. The target $y_j^{i,k}$ in task $(i, k)$is 0 or 1 which is sampled from the distribution $\mathbb{P}(y_j^{i,k} = 1 | x_j^{i,k} = x) = \frac{1}{1+\exp(\langle \theta^{i*}, x\rangle + b^{i*})}$, where θ^{i*}, b^{i*} is the true parameter and each entry of them (-1 or 1) are sampled with equal probability . In each task we generate n such data points. We use Equation (2) as our objective function, where we choose a loss function $l(y, \theta^T x + b) = (y - \frac{1}{1+\exp(\theta^T x + b)})^2$. This non-convex loss function has been used in (Mei et al., 2016; Nguyen & Sanner, 2013) and enjoys better accuracy in contrast to convex losses (e.g., logistic loss). NCGS, NCGS-VR, FW, SVFW and SVRG are compared in this experiment.

We choose different setting on m, K, n and report the results in Figure 2. In all experiment, the dimension of feature is set as $d = 300$ and we choose $R = 10$ in Equation (2).

We observe from Figure 2 that although we already adapt SVRG into the mini-batch version to speed up the converges, its convergence is still very slow due to large amount of computation in singular value decomposition when performing the projection onto the nuclear norm ball. We also observe that in all experiments, the proposed non-convex conditional gradient sliding methods outperfom their respective counterparts of the Frank-wolfe method. Particularly, NCGS-VR performs the best and is followed by NCGS , FW and SVFW. When the sample size is large (the right panel where the sample size is $30 \times 10 \times 3000$), our method is significantly better than the Frank-Wolfe method.

6.2. Real datasets

We test our algorithms on three real datasets: Aloi (n=108000, d=128) (Geusebroek et al., 2005), Covertype (n=581012, d=54) (Blackard & Dean, 1999) and Sensorless

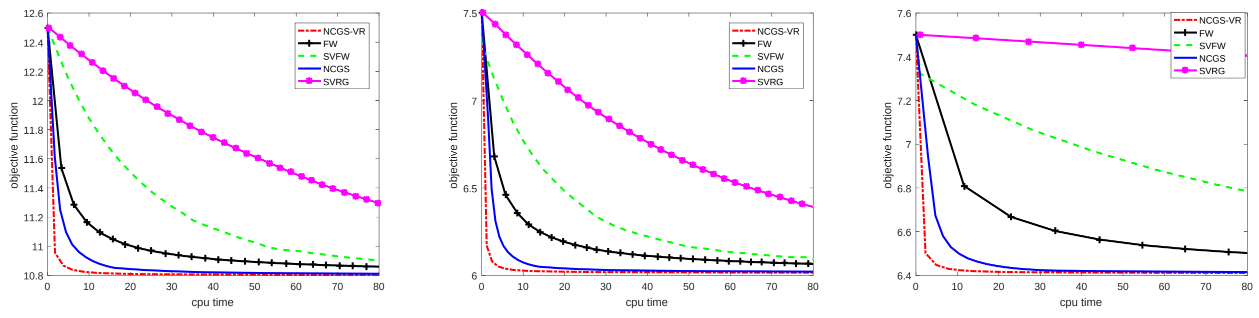

Figure 2. The X-axis is the cputime, the y-axis is the objective function. In the left figure, $m = 50$, $k = 5$, and $n = 1000$. In the middle figure, we have $m = 30$, $k = 5$, and $n = 1500$. In the right figure, $m = 30$, $k = 10$, and $n = 3000$.

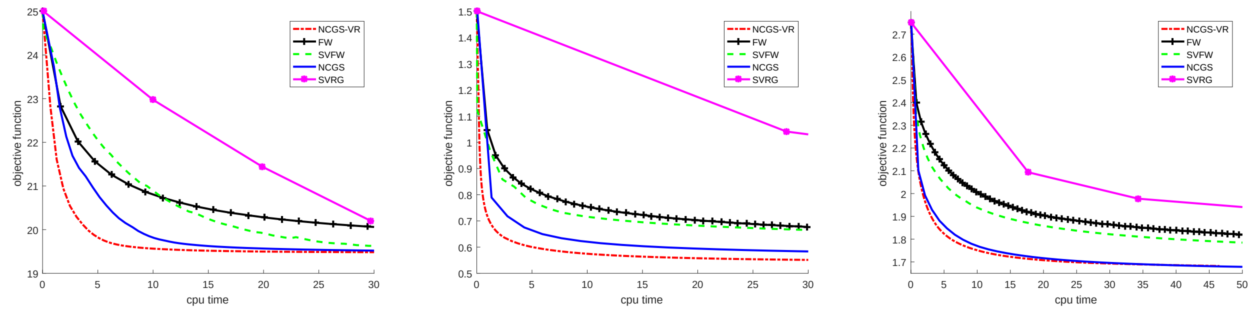

Figure 3. The X-axis is the cputime, the y-axis is the objective function. From the left to right, the dataset is aloi, covetype and Sensorless Drive Diagnosis.

Drive Diagnosis Data Set (n=58509, d=49) (Bator, 2015). We test the multitask learning task in Equation 2. For all dataset, we normalize the feature to the range $[-1, 1]$. Note these dataset have multi-classes. We generate tasks in the following way. For each class of a dataset, we generate five noisy versions of them by adding the small noise on the feature . Then we set the labels of these data to ones. We randomly sample same amount data from other classes and generate the noisy version of them in the same way, and then set the labels of them to zeros. Each version of data with label ones and zeros is one individual task in our multi-task learning problem. We report the experimental results in Figure 3. Same as before, we use the mini-batch version of SVRG.

In all experiments, SVRG converges very slowly due to heavy computation cost of the projection onto the nuclear norm ball. In the left figure, the performance of NCGS-VR is best and then is followed by NCGS. The non-convex Frank-Wolfe method converges fast at beginning then slow down, while SVFW performs in the opposite way. In the mid figure, NCGS-VR works fastest and is followed by NCGS, SVFW, and FW. In the right figure, the performance of NCGS-VR and NCGS are almost identical, and both outperform the counterpart of the Frank-Wolfe method.

7. Conclusion and future work

In this paper, we propose non-convex conditional gradient sliding methods to solve the batch, stochastic and finite-sum non-convex problems with complex constraints, such that projecting onto the feasible set is time consuming. Our algorithms surpass state of the art Frank-Wolfe type method both theoretically and empirically. One future research direction is to consider the accelerated steps in the proposed NCGS-VR algorithm, in hope to further improve the convergence speed.

References

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 699–707, 2016.

Allen-Zhu, Z. and Yuan, Y. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1080–1089, 2016.

Bator, M. UCI machine learning repository, 2015. URL `http://archive.ics.uci.edu/ml`.

Bertsekas, D. P. *Nonlinear programming*. Athena scientific Belmont, 1999.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Candes, E. J., Li, X., and Soltanolkotabi, M. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. Sparse and low-rank matrix decompositions. *IFAC Proceedings Volumes*, 42(10):1493–1498, 2009.

Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. L. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9(Aug): 1775–1822, 2008.

Dudik, M., Harchaoui, Z., and Malick, J. Lifted coordinate descent for learning with trace-norm regularization. In *Artificial Intelligence and Statistics*, pp. 327–336, 2012.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2): 95–110, 1956.

Garber, D. and Hazan, E. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *arXiv preprint arXiv:1301.4666*, 2013.

Garber, D. and Hazan, E. Faster rates for the frank-wolfe method over strongly-convex sets. In *ICML*, pp. 541–549, 2015.

Geusebroek, J.-M., Burghouts, G. J., and Smeulders, A. W. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Ghadimi, S. and Lan, G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Guzmán, C. and Nemirovski, A. On lower complexity bounds for large-scale smooth convex optimization. *Journal of Complexity*, 31(1):1–14, 2015.

Jaggi, M. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pp. 427–435, 2013.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Lacoste-Julien, S. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pp. 496–504, 2015.

Lan, G. The complexity of large-scale convex programming under a linear optimization oracle. department of industrial and systems engineering, university of florida, gainesville. Technical report, Florida. Technical Report, 2013.

Lan, G. and Zhou, Y. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.

Mairal, J., Ponce, J., Sapiro, G., Zisserman, A., and Bach, F. R. Supervised dictionary learning. In *Advances in neural information processing systems*, pp. 1033–1040, 2009.

Mei, S., Bai, Y., and Montanari, A. The landscape of empirical risk for non-convex losses. *arXiv preprint arXiv:1607.06534*, 2016.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nguyen, T. and Sanner, S. Algorithms for direct 0–1 loss optimization in binary classification. In *International Conference on Machine Learning*, pp. 1085–1093, 2013.

Pong, T. K., Tseng, P., Ji, S., and Ye, J. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 314–323, 2016a.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. Stochastic frank-wolfe methods for nonconvex optimization. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pp. 1244–1251. IEEE, 2016b.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. J. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems*, pp. 1145–1153, 2016c.

Shalev-Shwartz, S. Sdca without duality, regularization, and individual convexity. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 747–754, 2016.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Zhang, X., Schuurmans, D., and Yu, Y.-l. Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems*, pp. 2906–2914, 2012.