Fast Stochastic Algorithms for Low-rank and Nonsmooth Matrix Problems

Dan Garber
Technion - Israel Institute of Technology

Atara Kaplan
Technion - Israel Institute of Technology

Abstract

Composite convex optimization problems which include both a nonsmooth term and a low-rank promoting term have important applications in machine learning and signal processing, such as when one wishes to recover an unknown matrix that is simultaneously low-rank and sparse. However, such problems are highly challenging to solve in large-scale: the low-rank promoting term prohibits efficient implementations of proximal methods for composite optimization and even simple subgradient methods. On the other hand, methods which are tailored for low-rank optimization, such as conditional gradient-type methods, which are often applied to a smooth approximation of the nonsmooth objective, are slow since their runtime scales with both the large Lipchitz parameter of the smoothed gradient vector and with $1/\epsilon$, where ϵ is the target accuracy. In this paper we develop efficient algorithms for *stochastic* optimization of a strongly-convex objective which includes both a nonsmooth term and a low-rank promoting term. In particular, to the best of our knowledge, we present the first algorithm that enjoys all following critical properties for large-scale problems: i) (nearly) optimal sample complexity, ii) each iteration requires only a single *low-rank* SVD computation, and iii) overall number of thin-SVD computations scales only with $\log 1/\epsilon$ (as opposed to poly$(1/\epsilon)$ in previous methods). We also give an algorithm for the closely-related finite-sum setting. We empirically demonstrate our results on the problem of recovering a simultaneously low-rank and sparse matrix.

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

1 Introduction

Our paper is strongly motivated by low-rank and nonsmooth matrix optimization problems which are quite common in machine learning and signal processing applications. These include tasks such as *low-rank and sparse covariance matrix estimation*, *graph denoising* and *link prediction* [18], *analysis of social networks* [20], and *subspace clustering* [19], to name a few.

Such optimization problems often fit the following very general optimization model:

$$\min_{\mathbf{X}\in\mathbb{V}} f(\mathbf{X}) := G(\mathbf{X}) + R^{\mathrm{NS}}(\mathbf{X}) + h(\mathbf{X}), \qquad (1)$$

where $\mathbb{V}$ is a finite linear space over the reals, $G(\cdot)$ is convex and smooth, $R^{\mathrm{NS}}(\cdot)$ is convex and (generally) nonsmooth, and $h(\cdot)$ is convex and proximal-friendly (e.g., it is an indicator function for a convex set or a convex regularizer). Motivated by large-scale machine learning settings, we further assume $G(\cdot)$ is stochastic, i.e., $G(\mathbf{X}) = \mathbb{E}_{g\sim\mathcal{D}}[g(\mathbf{X})]$, where $\mathcal{D}$ is a distribution over convex and smooth functions, and either given by a sampling oracle (stochastic setting), or admits a finite support and given explicitly (finite-sum setting). Finally, we assume $f(\cdot)$ is strongly-convex (either due to strong convexity of $G(\cdot)$ or $R^{\mathrm{NS}}(\cdot)$). For instance the simultaneously low-rank and sparse covariance estimation problem [18] can be written as:

$$\min_{tr(\mathbf{X})\le\tau,\ \mathbf{X}\succeq 0} \frac{1}{2}\|\mathbf{X}-\mathbf{M}\|_F^2 + \lambda\|\mathbf{X}\|_1, \qquad (2)$$

where $\mathbf{M} = \mathbf{Y}\mathbf{Y}^\top + \mathbf{N}$ is a noisy observation of some low-rank and sparse covariance matrix $\mathbf{Y}\mathbf{Y}^\top$. Here, $\mathbb{V} = \mathbb{S}_n$ (space of $n\times n$ real symmetric matrices), $G(\mathbf{X}) = \frac{1}{2}\|\mathbf{X}-\mathbf{M}\|_F^2$ (which is deterministic in this simple example), $R^{\mathrm{NS}}(\mathbf{X}) = \lambda\|\mathbf{X}\|_1$, and $h(\mathbf{X})$ is an indicator function for the trace-bounded positive semidefinite cone (which both constraints the solution to be positive semidefinite and promotes low-rank)[1].

[1] A closely related problem to (2) to which all of the following discussions apply, is when $\mathbf{X}\in\mathbb{V} = \mathbb{R}^{m\times n}$ is not

The general model (1) is known to be a very difficult optimization problem to solve in large scale, already in the specific setting of Problem (2). In particular, many of the traditional first-order convex optimization methods used for solving non-smooth optimization problems are not efficiently applicable to it. For instance, proximal methods for composite optimization, such as the celebrated FISTA algorithm [1], do not admit efficient implementations for composite problems which include both a non-smooth term and a low-rank promoting term. When applied to Problem (2), each iteration of FISTA will require to solve a problem of the same form as the original problem, and hence is inefficient. Another type of well known first-order methods that are applicable to nonsmooth problems are deterministic/stochastic subgradient/mirror-descent methods [16, 3]. However these methods are also inefficient for problems such as (2), since each iteration requires projecting a point onto the feasible set, which requires a full-rank SVD computation on each iteration that is is computationally-prohibitive for large-scale problems.

Another type of methods, which are often suitable for large-scale low-rank matrix optimization problems, and have been studied extensively in this context in recent years, are Conditional Gradient-type methods (aka Frank Wolfe-type methods), see for instance [9, 5, 8, 7, 17, 14, 11, 10, 15, 6]. These type of algorithms, when applied to optimization over a nuclear-norm ball or over the trace-bounded positive semidefinite cone (as in Problem (2)), avoid expensive full-rank SVD computations, and only compute a single leading singular vector pair on each iteration (i.e., rank-one SVD), and hence are much more scalable. However, Conditional Gradient methods can usually be applied only to smooth problems, and so, the non-smooth term $R^{\mathrm{NS}}(\mathbf{X})$ is often replaced with a smooth approximation $R(\mathbf{X})$. A general theory and framework for generating such smooth approximation (i.e., replacing the non-smooth term with a smooth function that is pointwise close to the original), often referred to as *smoothing*, is described in [2]. Unfortunately, smoothing a function often results in a large Lipschitz constant of the gradient vector of the smoothed function. For example, the smooth approximation of the ℓ_1 norm is via the well known Huber function for which the Lipschitz constant of the gradient often scales like $\dim(\mathbb{V})/\varepsilon$, where ε is target accuracy to which the problem needs to be solved. Since the convergence rate of smooth optimization algorithms such as conditional gradient-type methods discussed above often scales with $\beta D^2/\epsilon$, where β is the Lipschitz parameter of the gradient and D is the distance of the initial point to an optimal solution, these methods are often not scalable for nonsmooth objectives such as Problem (2) and the general model (1) (even after smoothing them), since typically all three parameters $1/\varepsilon, D, \beta$ can be quite large. In particular, we note that for strongly-convex functions, it is possible to obtain (via other types of first-order methods) rates that depend only logarithmically on $1/\epsilon, D$.

Another issue with conditional gradient methods is that, as opposed to projected subgradient methods, their analysis does not naturally extend to handle stochastic objectives (recall that, motivated by machine learning settings, in the general model (1) we assume $G(\cdot)$ is stochastic). In particular, a straightforward variant of the method for stochastic objectives results in a highly suboptimal sample complexity [7]. In a recent related work [13], the authors consider a variant of the conditional gradient method for solving stochastic optimization problems that cleverly combines the conditional gradient method with Nesterov's accelerated method and stochastic sampling to obtain an algorithm for smooth stochastic convex optimization that, in the context of low-rank matrix optimization problems, i) requires only 1-SVD computation on each iteration (as in the standard conditional gradient method) and ii) enjoys (nearly) optimal sample complexity (both in the strongly convex case and non-strongly convex case). In a recent work [7], the technique of [13] was extended to the finite-sum stochastic setting and combined with a popular variance reduction technique [12], resulting in a conditional gradient-type method for smooth and strongly-convex finite-sum optimization that i) requires only 1-SVD computation on each iteration, and ii) enjoys a gradient-oracle complexity of the same flavor as usually obtained via variance-reduction methods [12], greatly improving over naive applications of conditional gradient methods which do not apply variance reduction. Unfortunately, both results [13, 7], while greatly improving the first-order oracle complexity of previous conditional-gradient methods, still require an overall number of 1-SVD computations that scales like $\beta D/\epsilon$. Hence, when applied to smooth approximations of nonsmooth problems such as Problems (2), (1), the overall very large number of thin-SVD computations needed greatly limits the applicability of these methods.

The limitations of previous methods in tackling large-scale low-rank and nonsmooth matrix optimization problems naturally leads us to the following question.

In the context of low-rank and nonsmooth matrix

constrained to be positive semidefinite (or even symmetric), and a low-rank solution is encouraged by constraining $\mathbf{X}$ via a nuclear norm constraint $\|\mathbf{X}\|_* \leq \tau$, where $\|\cdot\|_*$ is the ℓ_1 norm applied to the vector of singular values.

optimization, is it possible to combine all following three key properties for solving large-scale instances of Model (1) *into a single algorithm?*

1. *(nearly) optimal sample complexity,*
2. *use of only low-rank SVD computations,*
3. *overall number of low-rank SVD computations scales with* $\log(1/\varepsilon)$ *(not poly*$(1/\epsilon)$ *as in previous methods).*

In this paper we answer this question in the affirmative. To better discuss our results we now fully formalize the considered model and assumptions.

We consider the following general model:

$$\min_{\mathbf{X}\in\mathbb{V}} f(\mathbf{X}) := G(\mathbf{X}) + R(\mathbf{X}) + h(\mathbf{X}), \tag{3}$$

where $\mathbb{V}$ is a finite linear space over the reals equipped with an inner product $\langle\cdot,\cdot\rangle$. Throughout the paper we let $\|\cdot\|$ denote the norm induced by the inner product.

Throughout the paper we consider the following assumptions for model (3).

Assumption 1.
- *G is stochastic, i.e., $G(\mathbf{X}) = \mathbb{E}_{g\sim\mathcal{D}}[g(\mathbf{X})]$, where $\mathcal{D}$ is a distribution over functions $g : \mathbb{V} \to \mathbb{R}$, given by a sampling oracle. G is differentiable, and for all $g \in supp(\mathcal{D})$, g is β_G-smooth, and there exists $\sigma \geq 0$ such that $\sigma \geq \sup_{\mathbf{X}\in dom(h)} \sqrt{\mathbb{E}[\|\nabla G(\mathbf{X}) - \nabla g(\mathbf{X})\|^2]}$.*
- *$R : \mathbb{V} \to (-\infty, \infty]$ is deterministic, β_R-smooth, and convex,*
- *$G + R$ is α-strongly convex,*
- *$h : \mathbb{V} \to (-\infty, \infty]$ is deterministic, non-smooth, proper, lower semicontinuous and convex.*

For simplicity we define $\beta := \beta_G + \beta_R$. As discussed above, $R(\cdot)$ can be thought of as a smooth approximation of some nonsmooth term $R^{\text{NS}}(\cdot)$ (hence, we generally expect that $\beta_R >> \beta_G$), and $h(\cdot)$ can be thought of as either an indicator function for a convex set (e.g., a nuclear-norm ball) or a convex regularizer.

A quick summary of our results and comparison to previous conditional gradient-type methods for solving Model (3) in case $h(\cdot)$ is either an indicator for a nuclear norm ball of radius τ or the set of all positive semidefinite matrices with trace at most τ, is given in Table 1.

Our algorithm and novel complexity bounds are based on a combination of the variance reduction technique introduced in [12] and the use of, what we refer to in this work as, a *weak-proximal oracle* (as opposed to the standard exact proximal oracle used ubiquitously in first-order methods), which was introduced in the context of nuclear-norm-constrained optimization in [8], and further generalized in [17]. In the context of low-rank matrix optimization problems, implementation of this weak-proximal oracle requires a SVD computation of rank at most $\text{rank}(\mathbf{X}^*)$ - the rank of the optimal solution $\mathbf{X}^*$, as opposed to an exact proximal oracle that requires in general a full-rank SVD computation. Since for such problems we expect that $\text{rank}(\mathbf{X}^*)$ is much smaller than the dimension, and since the runtime of low-rank SVD computations (when carried out via fast iterative methods such as variants of the subspace iteration method or Lanczos-type algorithms) scales nicely with both the target rank and sparsity of gradients[2], for such problems the weak-proximal oracle admits a much more efficient implementation than the standard proximal oracle.

While both of these algorithmic ingredients are previously known and studied, it is their particular combination that, quite surprisingly, proves to be key to obtaining all three complexity bounds listed in our proposed question, simultaneously. In particular, it is important to note that while the use of a weak proximal oracle, as we define precisely in the sequel, suffices to obtain an algorithm that uses overall only $O(\log(1/\epsilon)$ *low-rank* SVD computations (currently treating for simplicity all other parameters as constants), to the best of our knowledge it does not suffice in order to also obtain (nearly) optimal sample complexity. The reason, at a high-level, is that the weak-proximal oracle is strong enough to guarantee decrease of the loss function on each iteration (in expectation), but does not give a stronger type of guarantee, which holds for the exact proximal oracle, that is crucial for obtaining optimal sample complexity with algorithms such as Stochastic Gradient Descent [3] and the conditional gradient-type method of [13] (that indeed rely on exact, or nearly exact, proximal computations). It turns out that the use of a variance reduction technique (such as [12]) is key to bypassing this obstacle and obtaining also (near) optimal sample complexity, on top of the low SVD complexity. We also give a variant of our algorithm to the finite-sum setting that obtains similar improvements.

Finally, while our main motivation comes from low-rank and nomsmooth matrix optimization problems, it is important to note that as captured in our general Model (3), our results are applicable in a much wider setting than that of low-rank matrix optimization problems. Our method is suitable especially for stochastic nonsmooth convex problems for which im-

[2] see for instance discussions in [8].

Algorithm	#Exact Gradients	#Stochastic Gradients	SVD rank	#SVD Computations
↓ Stochastic Setting ↓				
Stochastic Cond. Grad. [7]	0	$\frac{\sigma^2\beta\tau^4}{\varepsilon^3}$	1	$\frac{\beta\tau^2}{\varepsilon}$
CGS [13]	0	$\frac{\sigma^2}{\alpha\varepsilon}$	1	$\frac{\beta\tau^2}{\varepsilon}$
This work (Alg. 1)	0	$\frac{\sigma^2}{\alpha\varepsilon}$	$\text{rank}(\mathbf{X}^*)$	$\frac{\beta}{\alpha}\ln\left(\frac{1}{\varepsilon}\right)$
↓ Finite Sum ↓				
STORC [7]	$\ln\left(\frac{1}{\varepsilon}\right)$	$\left(\frac{\beta}{\alpha}\right)^2\ln\left(\frac{1}{\varepsilon}\right)$	1	$\frac{\beta\tau^2}{\varepsilon}$
This work finite sum (Alg. 2)	$\ln\left(\frac{1}{\varepsilon}\right)$	$\frac{\beta_G^2\beta}{\alpha^3}\ln\left(\frac{1}{\varepsilon}\right)$	$\text{rank}(\mathbf{X}^*)$	$\frac{\beta}{\alpha}\ln\left(\frac{1}{\varepsilon}\right)$

Table 1: Comparison of complexity bound for conditional gradient-type methods for solving Model (3). $\mathbf{X}^*$ denotes the unique optimal solution. Table only lists the leading-order terms.

plementing a weak proximal oracle is much more efficient than an exact proximal oracle.

2 Algorithm and Results

Our algorithm for solving Model (3), Algorithm 1, is given below. We now briefly discuss the main two building blocks of the algorithm, namely a variance reduction technique and the use of a weak proximal oracle.

Our use of the variance reduction technique of [12] is quite straightforward as observable in Algorithm 1. Importantly, while [12] applied it to finite-sum optimization, here we apply it to the more general black-box stochastic setting, and hence the sample-size parameter k_s used for the "snap-shot" gradient $\tilde{\nabla}g(\mathbf{X}_s)$ on epoch s grows from epoch to epoch. This modification of the technique is along the lines of [4].

The weak proximal oracle strategy is applied in our algorithm as follows. For a step-size η_t, a composite optimization proximal algorithm, which treats the function $h(\cdot)$ in proximal fashion and the functions G, R via a gradient oracle, will compute on each iteration an update of the form

$$\mathbf{V}_t \leftarrow \arg\min_{\mathbf{V}\in\mathbb{V}}\Big\{\psi_t(\mathbf{V}) := \frac{1}{\beta\eta_t}h(\mathbf{V}) + \|\mathbf{V} - \mathbf{X}_{s,t}\frac{1}{2\beta\eta_t}(\hat{\nabla}g(\mathbf{X}_{s,t}) + \nabla R(\mathbf{X}_{s,t}))\|^2 + \Big\}. \quad (4)$$

For instance, if $\mathbb{V} = \mathbb{R}^{m\times n}$ and $h(\cdot)$ is an indicator function for the nuclear-norm ball $\{\mathbf{X} \in \mathbb{R}^{m\times n} \mid \|\mathbf{X}\|_* \le \tau\}$, then computing $\mathbf{V}_t$ in Eq. (4) amounts to Euclidean projection of the matrix $\mathbf{A}_t = \mathbf{X}_{s,t} - \frac{1}{2\beta\eta_t}(\hat{\nabla}g(\mathbf{X}_{s,t}) - \nabla R(\mathbf{X}_{s,t}))$ onto the nuclear-norm ball of radius τ. This projection is carried out by computing a full-rank SVD of $\mathbf{A}_t$ and projecting the singular values onto the τ-scaled simplex. Since a full-rank SVD is required, this operation takes $O(m^2n)$ time (assuming $m \le n$), which is prohibitive for very large m, n.

Our algorithm avoids the computational bottleneck of full-rank SVD computations by only requiring that $\mathbf{V}_t$ satisfies the inequality:

$$\psi_t(\mathbf{V}_t) \le \psi_t(\mathbf{X}^*), \quad (5)$$

where $\mathbf{X}^*$ is the (unique) optimal solution to (3). We call a procedure for computing such updates - a weak proximal oracle. In the context discussed above, i.e., $h(\cdot)$ is an indicator for the radius-τ nuclear-norm ball, (5) can be satisfied simply by projecting the $\text{rank}(\mathbf{X}^*)$-approximation of the matrix $\mathbf{A}_t$ onto the nuclear-norm ball. This only requires to compute the top $\text{rank}(\mathbf{X}^*)$ components in the singular value decomposition of $\mathbf{A}_t$, and thus the runtime scales roughly like $O(\text{rank}(\mathbf{X}^*)\cdot\text{nnz}(\mathbf{A}_t))$ using fast Krylov Subspace methods (e.g., subspace iteration, Lanczos), which results in a much more efficient procedure (see further detailed discussions in [8, 17]).

Since in many settings of interest, especially in the context of matrix optimization problems, the computation of $\mathbf{V}_t$ requires some numeric procedure which is prone to accuracy issues, or in cases in which $\mathbf{X}^*$ is not low-rank but only very close to a low-rank matrix (in some norm), we introduce an error-tolerance parameter δ in the proximal computation step in Algorithm 1 which allows to absorb such errors that can be controlled (e.g., by properly tuning precision of the thin-SVD computation).

2.1 Outline of main results

Theorem 1 (stochastic setting)**.** *Assume that Assumption 1 holds. There is an explicit choice for the parameters in Algorithm 1 for which the total number of epochs (iterations of the outer-loop) required in order to find an ε-approximated solution in expectation for Problem (3) is bounded by $O\left(\ln\left(\frac{1}{\varepsilon}\right)\right)$, the total number of calls to the weak proximal oracle is bounded by $O\left(\frac{\beta}{\alpha}\ln\left(\frac{1}{\varepsilon}\right)\right)$, and the total number of stochastic*

Algorithm 1 Stochastic Variance-Reduced Generalized Conditional Gradient for Problem (3)

Input: T, $\{\eta_t\}_{t=1}^{T-1} \subset [0,1]$, $\{k_t\}_{t=1}^{T-1}, \{k_s\}_{s\geq 1} \subset \mathbb{N}$, $\delta \geq 0$.
Initialization: Choose some $\mathbf{X}_1 \in dom(h)$.
for $s = 1, 2, ...$ **do**
  Sample $g^{(1)}, ..., g^{(k_s)}$ from $\mathcal{D}$.
  $\tilde{\nabla} g(\mathbf{X}_s) = \frac{1}{k_s}\sum_{i=1}^{k_s} \nabla g^{(i)}(\mathbf{X}_s)$ {snap-shot gradient} {in the finite-sum setting we use $\tilde{\nabla} g(\mathbf{X}_s) = \frac{1}{n}\sum_{i=1}^{n} \nabla g^{(i)}(\mathbf{X}_s)$}.
  $\mathbf{X}_{s,1} = \mathbf{X}_s$
  for $t = 1, 2, ..., T-1$ **do**
    Sample $g^{(1)}, ..., g^{(k_t)}$ from $\mathcal{D}$.
    $\hat{\nabla} g(\mathbf{X}_{s,t}) = \frac{\sum_{i=1}^{k_t}\left(\nabla g^{(i)}(\mathbf{X}_{s,t}) - \left(\nabla g^{(i)}(\mathbf{X}_s) - \tilde{\nabla} g(\mathbf{X}_s)\right)\right)}{k_t}$.
    $\mathbf{V}_t = \arg\min_{\mathbf{V}\in\mathbb{V}} \psi_t(\mathbf{V})$ (see Eq. (4)) {in fact it suffices that $\psi_t(\mathbf{V}_t) \leq \psi_t(\mathbf{X}^*) + \delta$ for some optimal solution $\mathbf{X}^*$}.
    $\mathbf{X}_{s,t+1} = (1-\eta_t)\mathbf{X}_{s,t} + \eta_t \mathbf{V}_t$
  end for
  $\mathbf{X}_{s+1} = \mathbf{X}_{s,T}$
end for

gradients sampled is bounded by $O\left(\frac{\sigma^2}{\alpha\varepsilon} + \frac{\beta_G^2\beta}{\alpha^3}\ln\left(\frac{1}{\varepsilon}\right)\right)$.

We note that under Assumption 1, the overall number of calls to a weak proximal oracle to reach ϵ-approximated solution matches the overall number of calls to an *exact* proximal oracle used by the proximal gradient method for smooth and strongly convex optimization. Also, under Assumption 1, the leading term in the bound on overall number of stochastic gradients is optimal (up to constants).

Theorem 2 (finite-sum setting)**.** *Assume that Assumption 1 holds and that $\mathcal{D}$ is an explicitly given uniform distribution over n functions. There exist an explicit choice for the parameters in Algorithm 2 (see appendix) for which the total number of epochs required in order to find an ε-approximated solution in expectation for Problem* (3) *is bounded by $O\left(\ln\left(\frac{1}{\varepsilon}\right)\right)$, the total number of calls to the weak proximal oracle is bounded by $O\left(\frac{\beta}{\alpha}\ln\left(\frac{1}{\varepsilon}\right)\right)$, and the total number of gradients computed for any of the n functions in the support of $\mathcal{D}$ is bounded by $O\left(\left(n + \frac{\beta_G^2\beta}{\alpha^3}\right)\ln\left(\frac{1}{\varepsilon}\right)\right)$.*

We see that as is standard in variance-reduced methods for smooth and strongly convex optimization, the overall number of gradients decouples between terms that depend on the smoothness and strong convexity of the objective (e.g., the condition number β/α), and the overall number of functions n.

3 Analysis

Due to lack of space most of the proofs and formal arguments are deferred to the appendix. Here we outline the main steps in proving Theorem 1. The treatment for Theorem 2 is very similar and given in the appendix.

The following lemma bounds the expected decrease in function value after a single iteration of the inner-loop in Algorithm 1. The proof relies on the smoothness and strong convexity of $G + R$, the use of the weak-proximal oracle and the unbiased gradient estimator.

Lemma 1 (expected decrease)**.** *Assume that Assumption 1 holds. Fix some epoch s of Algorithm 1, and let $\{\mathbf{X}_{s,t}\}_{t=1}^{T+1}$, $\{\mathbf{V}_t\}_{t=1}^{T}$ be the iterates generated throughout the epoch, and suppose that $\psi_t(\mathbf{V}_t) \leq \psi_t(\tilde{\mathbf{X}}) + \delta$ for some fixed feasible solution $\tilde{\mathbf{X}}$. Then, if $2\beta\eta_t \leq \alpha$, we have that $\mathbb{E}[f(\mathbf{X}_{s,t+1})] \leq (1-\eta_t)\,\mathbb{E}[f(\mathbf{X}_{s,t})] + \eta_t f(\tilde{\mathbf{X}}) + \frac{\sigma_{s,t}^2}{2\beta} + \beta\eta_t^2\delta$, where $\sigma_{s,t} = \sqrt{\mathbb{E}[\|\nabla G(\mathbf{X}_{s,t}) - \hat{\nabla} g(\mathbf{X}_{s,t})\|^2]}$.*

The following lemma bounds the variance the gradient estimator used in any iteration of the inner-loop of Algorithm 1. The proof is based on the smoothness of functions in the support of $\mathcal{D}$ and strong-convexity of $G + R$.

Lemma 2 (variance bound)**.** *Assume that Assumption 1 holds. Fix some epoch s of Algorithm 1, and let $\{\mathbf{X}_{s,t}\}_{t=1}^{T+1}$ be the iterates generated throughout the epoch. Then, $\sigma_{s,t}^2 = \mathbb{E}[\|\nabla G(\mathbf{X}_{s,t}) - \hat{\nabla} g(\mathbf{X}_{s,t})\|^2] \leq \frac{8\beta_G^2}{\alpha k_t}(\mathbb{E}[f(\mathbf{X}_s)] - f(\mathbf{X}^*)) + \frac{8\beta_G^2}{\alpha k_t}(\mathbb{E}[f(\mathbf{X}_{s,t})] - f(\mathbf{X}^*)) + \frac{2\sigma^2}{k_s}$.*

The following theorem, from which Theorem 1 (and with slight changes also Theorem 2) essentially follows, bounds the approximation error of Algorithm 1.

Theorem 3. *Assume that Assumption 1 holds. Let $\{\mathbf{X}_s\}_{s\geq 1}$ be a sequence generated by Algorithm 1 with parameters $T = \frac{8\beta}{3\alpha}\ln 8 + 1$, $\eta_t = \frac{\alpha}{2\beta}$, $k_s = \frac{32\sigma^2}{\alpha C_0}2^{s-1}$ and $k_t = \frac{32\beta_G^2}{\alpha^2}$, where $C_0 \geq h_1$. Then, for all $s \geq 1$ it holds that: $\mathbb{E}[f(\mathbf{X}_s)] - f(\mathbf{X}^*) \leq C_0\left(\frac{1}{2}\right)^{s-1} + \frac{8\alpha\delta}{7}$.*

Proof. Let us define $h_s := \mathbb{E}[f(\mathbf{X}_s)] - f(\mathbf{X}^*)$ for all $s \geq 1$, and $h_{s,t} := \mathbb{E}[f(\mathbf{X}_{s,t})] - f(\mathbf{X}^*)$ for all $s, t \geq 1$. Fix some epoch s and iteration t of the inner loop.

Using Lemma 1 with $\tilde{\mathbf{X}} = \mathbf{X}^*$, and Lemma 2 we have that

$$h_{s,t+1} \leq (1-\eta_t)\,h_{s,t} + \frac{1}{2\beta}\left(\frac{8\beta_G^2}{\alpha k_t}h_s + \frac{8\beta_G^2}{\alpha k_t}h_{s,t} + \frac{2\sigma^2}{k_s}\right) + \beta\eta_t^2\delta$$
$$= \left(1 - \eta_t + \frac{4\beta_G^2}{\alpha\beta k_t}\right)h_{s,t} + \left(\frac{4\beta_G^2}{\alpha\beta k_t}h_s + \frac{\sigma^2}{\beta k_s} + \beta\eta_t^2\delta\right).$$

Plugging $k_t = \frac{16\beta_G^2}{\alpha\beta\eta_t}$ we get

$$h_{s,t+1} \leq \left(1 - \eta_t + \frac{\eta_t}{4}\right) h_{s,t} + \left(\frac{\eta_t}{4} h_s + \frac{\sigma^2}{\beta k_s} + \beta\eta_t^2\delta\right).$$

Plugging $\eta_t = \frac{\alpha}{2\beta}$ we get

$$h_{s,t+1} \leq \left(1 - \frac{3\alpha}{8\beta}\right) h_{s,t} + \left(\frac{\alpha}{8\beta} h_s + \frac{\sigma^2}{\beta k_s} + \frac{\alpha^2\delta}{4\beta}\right).$$

Fixing an epoch s and unrolling the recursion for $t = (T-1)\dots 1$ we get

$$\begin{aligned}
h_{s,T} &\leq \left(1 - \frac{3\alpha}{8\beta}\right) h_{s,T-1} + \left(\frac{\alpha}{8\beta} h_s + \frac{\sigma^2}{\beta k_s} + \frac{\alpha^2\delta}{4\beta}\right) \\
&\leq \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1} h_{s,1} \\
&\quad + \left(\frac{\alpha}{8\beta} h_s + \frac{\sigma^2}{\beta k_s} + \frac{\alpha^2\delta}{4\beta}\right) \sum_{k=1}^{T-1} \left(1 - \frac{3\alpha}{8\beta}\right)^{T-k-1} \\
&= \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1} h_{s,1} \\
&\quad + \left(\frac{1}{3} h_s + \frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}\right) \left(1 - \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1}\right).
\end{aligned}$$

$h_{s,T} = h_{s+1}$ and $h_{s,1} = h_s$ and so

$$\begin{aligned}
h_{s+1} &\leq \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1} h_s \\
&\quad + \left(\frac{1}{3} h_s + \frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}\right) \left(1 - \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1}\right) \\
&= \left(\frac{1}{3} + \frac{2}{3}\left(1 - \frac{3\alpha}{8\beta}\right)^{T-1}\right) h_s \\
&\quad + \left(\frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}\right) \left(1 - \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1}\right) \\
&\leq \left(\frac{1}{3} + \frac{2}{3} e^{-\frac{3\alpha}{8\beta}(T-1)}\right) h_s \\
&\quad + \left(\frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}\right) \left(1 - \left(1 - \frac{3\alpha}{8\beta}\right)^{T-1}\right).
\end{aligned}$$

Choosing $T = \frac{8\beta}{3\alpha}\ln 8 + 1$, we get

$$\begin{aligned}
h_{s+1} &\leq \left(\frac{1}{3} + \frac{2}{3} e^{-\frac{3\alpha}{8\beta}\left(\frac{8\beta}{3\alpha}\ln 8\right)}\right) h_s \\
&\quad + \left(\frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}\right) \left(1 - \left(1 - \frac{3\alpha}{8\beta}\right)^{\frac{8\beta}{3\alpha}\ln 8}\right) \\
&\leq \frac{5}{12} h_s + \frac{8\sigma^2}{3\alpha k_s} + \frac{2\alpha\delta}{3}.
\end{aligned}$$

Finally, plugging the value of k_s, the result follows from a simple induction over s (see appendix for complete argument) □

4 Applications to Non-smooth Problems

4.1 Applying our results to non-smooth problems via smoothing

In order to fit the nonsmooth problems considered in this section to our smooth model (3), we build on the smoothing framework introduced in [2], which replaces the nonsmooth term $R(\mathbf{X})$ with a smooth approximation.

The following definition is taken from [2].

Definition 1. *Let $R : \mathbb{V} \to (-\infty, \infty]$ be a closed, proper and convex function and let $X \subseteq dom(R)$ be a closed and convex set. R is (θ, γ, K)-smoothable over X if there exists γ_1 and γ_2 such that $\gamma = \gamma_1 + \gamma_2 \geq 0$ such that for every $\mu \geq 0$ there exists a continuously differentiable function $R_\mu : \mathbb{V} \to (-\infty, \infty]$ such that:*

(a) $R(x) - \gamma_1\mu \leq R_\mu(x) \leq R(x) + \gamma_2\mu$ for every $x \in X$.

(b) There exists $K \geq 0$ and $\theta \geq 0$ such that $\|\nabla R_\mu(x) - \nabla R_\mu(y)\| \leq \left(K + \frac{\theta}{\mu}\right)\|x - y\|$ for every $x, y \in X$.

Formally, now we consider applying our algorithms to non-smooth optimization problems of the following form:

$$\min_{\mathbf{X}\in\mathbb{V}} f(\mathbf{X}) := G(\mathbf{X}) + R(\mathbf{X}) + h(\mathbf{X}), \tag{6}$$

with the following assumptions.

Assumption 2. *We make the same assumptions as in Assumption 1 with the single difference that now R need not be smooth, but only (θ, γ, K)-smoothable.*

We will denote the μ-smooth approximation of $R(\mathbf{X})$ as $R_\mu(\mathbf{X})$, and its smoothness parameter to be $\beta_R = \left(K + \frac{\theta}{\mu}\right)$.

As in our discussions so far, considering Model (6) especially in the context of low-rank matrix optimization problems (e.g., $h(\cdot)$ is an indicator function for a nuclear-norm ball), we assume that the optimal solution $\mathbf{X}^*$ is naturally of low-rank and we want to rely on SVD computations whose rank does not exceeds that of $\mathbf{X}^*$ - the optimal solution to the *original non-smooth problem*. However, the rank of SVD computations required by the results developed in previous sections corresponded to the optimal solution of

the *smoothed problem*, i.e., after $R(\cdot)$ is replaced with a smooth approximation $R_\mu(\cdot)$, which can be higher. Thus, towards developing an algorithm that relies on SVD computation with rank at most that of the *non-smooth optimum*, we introduce the following modified definition of a weak-proximal oracle.

Definition 2. *We say an Algorithm $\mathcal{A}$ is a (δ_1, δ_2)-weak proximal oracle for Model (6),if for point $\mathbf{X} \in dom(h)$ and step-size η, $\mathcal{A}(\mathbf{X}, \eta)$ returns a point $\mathbf{V} \in dom(h)$ such that $\psi(\mathbf{V}, \mathbf{X}, \eta) \leq \psi(\tilde{\mathbf{X}}^*, \mathbf{X}, \eta) + \delta_1$, where $\tilde{\mathbf{X}}^*$ is a feasible point satisfying $|f(\mathbf{X}^*) - f(\tilde{\mathbf{X}}^*)| \leq \delta_2$, $\psi(\mathbf{V}, \mathbf{X}, \eta) := \|\mathbf{V} - \mathbf{X} + \frac{1}{2\beta\eta_t}(\hat{\nabla} g(\mathbf{X}) + \nabla R_\mu(\mathbf{X}))\|^2 + \frac{1}{\beta\eta_t} h(\mathbf{V})$, and $R_\mu(\cdot)$ is the μ-smooth approximation of $R(\cdot)$.*

Henceforth, we consider Algorithm 1 with the single difference: now $\mathbf{V}_t$ is the ouput of a (δ_1, δ_2)-weak proximal oracle, as defined in Definition 2. Note that in the context of low-rank problems and in the ideal case $\delta_1 = \delta_2 = 0$[3], the implementation of the oracle in Definition 2 is exactly the same as the weak proximal oracle discussed before, i.e., if $h(\cdot)$ is for instance the indicator function for a radius-τ nuclear-norm ball, then implementing the oracle in Definition 2 amounts to a Euclidean projection of the rank($\mathbf{X}^*$)-approximation of $\mathbf{A}_t := \mathbf{X} - \frac{1}{2\beta\eta_t}(\hat{\nabla} g(\mathbf{X}) + \nabla R_\mu(\mathbf{X}))$ onto the nuclear-norm ball. Here, the tolerances δ_1, δ_2 allow us to absorb the error due to the smoothing approximation and numerical errors in SVD computations.

Corollary 1. *Assume that Assumption 2 holds. Choosing parameters $\delta_1 = \frac{7\varepsilon}{32\alpha}$ and $\mu = \frac{7\varepsilon}{92\gamma}$, guarantees that the overall number of epochs to reach an ϵ-approximated solution in expectation is bounded by $O\left(\ln\left(\frac{1}{\varepsilon}\right)\right)$, the total number of calls to the (δ_1, δ_2)-weak proximal oracle is bounded by $O\left(\frac{\beta}{\alpha}\ln\left(\frac{1}{\varepsilon}\right)\right)$, and the total number of stochastic gradients sampled is bounded by $O\left(\frac{\sigma^2}{\alpha\varepsilon} + \frac{\beta_G^2\beta}{\alpha^3}\ln\left(\frac{1}{\varepsilon}\right)\right)$.* [4]

4.2 Specific examples

4.2.1 Low-rank and sparse matrix estimation

As discussed in the introduction, this work is largely motivated by matrix recovery problems, such as the following low-rank and sparse matrix estimation.

$$\min_{\|\mathbf{X}\|_* \leq \tau} \frac{1}{2}\|\mathbf{X} - \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}]\|_F^2 + \lambda\|\mathbf{X}\|_1, \qquad (7)$$

where $\mathcal{D}$ is an unknown distribution over instances.

[3] these can be made arbitarily small by the choice of smoothing parameter and accuracy in SVD computations.

[4] Recall that in this section $\beta = \beta_G + \beta_R = \beta_G + K + \theta/\mu$, which will typically scale with $1/\epsilon$ (inverse of desired approximation error). See following examples.

For problem (7) to fit the Model (6), we take $G(\mathbf{X}) = \mathbb{E}_{(\mathbf{M},\mathbf{N})\sim\mathcal{D}\times\mathcal{D}}\left[\frac{1}{2}\langle \mathbf{X} - \mathbf{M}, \mathbf{X} - \mathbf{N}\rangle\right]$. Since $\mathbf{M}$ and $\mathbf{N}$ are i.i.d, this is equivalent to

$$\begin{aligned} G(\mathbf{X}) &= \frac{1}{2}\langle \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{X} - \mathbf{M}], \mathbb{E}_{\mathbf{N}\sim\mathcal{D}}[\mathbf{X} - \mathbf{N}]\rangle \\ &= \frac{1}{2}\langle \mathbf{X} - \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}], \mathbf{X} - \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}]\rangle \\ &= \frac{1}{2}\|\mathbf{X} - \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}]\|_F^2. \end{aligned}$$

Smoothing the ℓ_1-norm has a well known solution, as shown in [2]. The μ-smooth approximation of $\|\mathbf{X}\|_1$ is $R_\mu(\mathbf{X}) = \sum_{j=1}^d \sum_{i=1}^m H_\mu(\mathbf{X}_{ij})$, with parameters $(1, \frac{md}{2}, 0)$, where $H_\mu(t)$ is the one dimensional Huber function:

$$H_\mu(t) = \begin{cases} \frac{t^2}{2\mu}, & |t| \leq \mu \\ |t| - \frac{\mu}{2}, & |t| > \mu \end{cases}.$$

This satisfies $R_\mu(\mathbf{X}) \leq \|\mathbf{X}\|_1 \leq R_\mu(\mathbf{X}) + \frac{md\mu}{2}$.

4.2.2 Linearly constrained low-rank matrix estimation

Another example, is the problem of recovering a low-rank matrix subject to linear constraints, which can be written in penalized form as:

$$\min_{\|\mathbf{X}\|_* \leq \tau} \frac{1}{2}\|\mathbf{X} - \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}]\|_F^2 + \max_{i\in[n]}\left(\langle \mathbf{A}_i, \mathbf{X}\rangle - b_i\right), \qquad (8)$$

where $\mathcal{D}$ is again an unknown distribution over instances. Here the matrices $\{\mathbf{A}_i\}_{i\in[n]}$ and scalars $\{b_i\}_{i\in[n]}$ can absorb a penalty factor λ.

Here, by [2], the μ-smooth approximation of $\max_{i\in[n]}\left(\langle \mathbf{A}_i, \mathbf{X}\rangle - b_i\right)$ is $R_\mu(\mathbf{X}) = \mu\log\left(\sum_{i=1}^n e^{\frac{1}{\mu}(\langle \mathbf{A}_i, \mathbf{X}\rangle - b_i)}\right)$, with parameters $(\|\mathcal{A}\|^2, \log n, 0)$, where $\mathcal{A} : \mathbb{R}^{m\times d} \to \mathbb{R}^n$ is a linear transformation with the form $\mathcal{A}(\mathbf{X}) = \left(\ tr(\mathbf{A}_1^T\mathbf{X}),\ tr(\mathbf{A}_2^T\mathbf{X}),\ \ldots,\ tr(\mathbf{A}_n^T\mathbf{X})\ \right)^\top$, for $\mathbf{A}_1, ..., \mathbf{A}_n \in \mathbb{R}^{m\times d}$, and $\|\mathcal{A}\| = \max\{\|\mathcal{A}(\mathbf{X})\|_2 : \|\mathbf{X}\|_F = 1\}$. This satisfies $R_\mu(\mathbf{X}) \leq \max_{i\in[n]}\left(\langle \mathbf{A}_i, \mathbf{X}\rangle - b_i\right) \leq R_\mu(\mathbf{X}) + \mu\log n$.

4.2.3 Low-rank matrix sensing with Elastic-net

Finally, we very briefly discuss a matrix-sensing problem, where both a nuclear-norm constraint is used to promote low-rank solutions and the well known elastic-net regularizer [21] is used to promote sparsity.

$$\min_{\|\mathbf{X}\|_* \leq \tau} \mathbb{E}_{(\mathbf{A},b)\sim\mathcal{D}}\left[\frac{1}{2}\left(\langle \mathbf{A}, \mathbf{X}\rangle - b\right)^2\right] + \lambda_1\|\mathbf{X}\|_1 + \lambda_2\|\mathbf{X}\|_F^2.$$

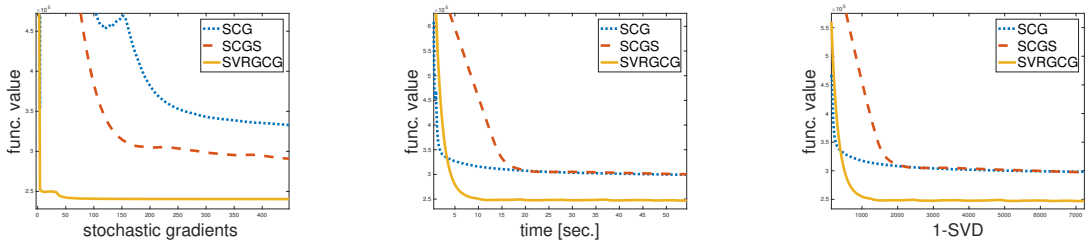

Figure 1: Comparison between methods with $\text{rank}(\mathbf{Y}\mathbf{Y}^\top) = 1$.

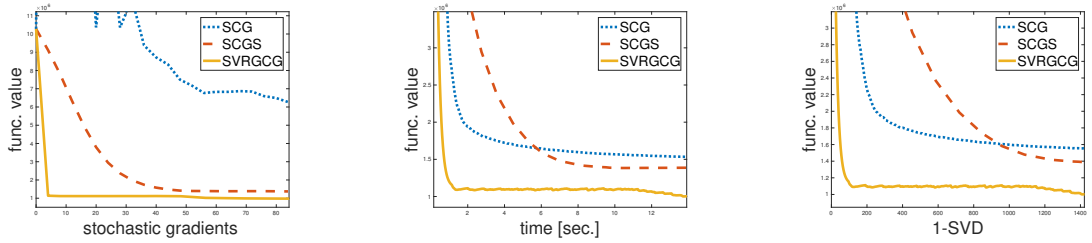

Figure 2: Comparison between methods with $\text{rank}(\mathbf{Y}\mathbf{Y}^\top) = 10$.

In this example, $G(\mathbf{X}) = \mathbb{E}_{(\mathbf{A},b)\sim\mathcal{D}}\left[\frac{1}{2}\left(\langle \mathbf{A}, \mathbf{X}\rangle - b\right)^2\right]$ need not be strongly convex as in previous examples, however the elastic-net regularizer $R(\mathbf{X}) := \lambda_1\|\mathbf{X}\|_1 + \lambda_2\|\mathbf{X}\|_F^2$ is strongly convex. The smoothing of in this example and resulting application of our method goes along the same lines as our treatment of Problem (7).

5 Experiments

In support of our theory, we present preliminary empirical experiments on the problem of *low-rank and sparse matrix estimation*, Problem (7). We compare our Algorithm 1 (SVRGCG) to previous conditional gradient-type stochastic methods including the Stochastic Conditional Gradient Algorithm (SCG) [7][5] and the Stochastic Conditional Gradient Sliding Algorithm (SCGS) [13]. We use synthetic randomly-generated data for the experiments. For all experiments the input matrix is of the form $\mathbf{M}_0 = \mathbb{E}_{\mathbf{M}\sim\mathcal{D}}[\mathbf{M}] = \mathbf{Y}\mathbf{Y}^\top + \mathbf{N}$, where $\mathbf{Y} \in \mathbb{R}^{d\times r}$ is a random sparse matrix for which each entry is zero w.p. $1 - 1/\sqrt{d}$ and $U\{1,\ldots,10\}$ w.p. $1/\sqrt{d}$, $\mathbf{N}$ is a $d \times d$ random matrix with i.i.d. standard Gaussian entries. We set the dimension to $d = 300$ and the rank of $\mathbf{Y}$, r to either 1 or 10. In all experiments we set $\lambda = 2$, $\varepsilon = 0.01\cdot\|\mathbf{Y}\mathbf{Y}^\top\|_F^2$ (i.e., the approximation error is relative to magnitude of signal), $\mu = \varepsilon/d^2$ (in accordance with Corollary 3) , and $\tau = \text{Tr}(\mathbf{Y}\mathbf{Y}^\top)$. The stochastic oracle is implemented by taking noisy observations of $\mathbf{M}_0$ using: $\mathbf{M}^{(i)} = \mathbf{M}_0 + \sigma\mathbf{Q}^{(i)}$, where each $\mathbf{Q}^{(i)}$ is random with i.i.d. standard Gaussian entries and we fix $\sigma = 5$.

[5] In [7] it appears as Stochastic Frank-Wolfe (SFW).

For all three methods we measure i) the obtained (original non-smooth) function value (see (7)) vs. number of stochastic gradients used, ii) function value vs. overall runtime (seconds), and iii) function value vs. overall number of rank-one SVD computations used. Since the overall running time is highly dependent on specific implementation, we bring the number of rank-one SVD computations as an implementation-independent proxy for the overall runtime. For our method SVRGCG, we compute the overall number of rank-one SVD computations by multiplying the number of SVD factorizations with the rank of the factorization used[6]. In the first experiment (Figure 1) we set $\text{rank}(\mathbf{Y}) = 1$, and in a second experiment (Figure 2), we set $\text{rank}(\mathbf{Y}) = 10$ in which case, our algorithm SVRGCG uses rank-10 SVD computations. The results for each experiment are averages of 30 i.i.d. runs. All three algorithms were implemented using parameters as suggested by theory without attempts to optimize their performance.

In Figures 1,2, it can be seen that our algorithm SVRGCG clearly outperforms both SCG and SCGS with respect to all three measures in the two experiments.

6 Acknowledgments

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 1108/18).

[6] This is reasonable since the runtime for low-rank SVD typically scales linearly with rank.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[2] Amir Beck and Marc Teboulle. Smoothing and first order methods: a unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[3] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[4] Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, pages 728–763, 2015.

[5] Dan Garber. Faster projection-free convex optimization over the spectrahedron. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 874–882, 2016.

[6] Donald Goldfarb, Garud Iyengar, and Chaoxu Zhou. Linear convergence of stochastic frank wolfe variants. In *Artificial Intelligence and Statistics*, pages 1066–1074, 2017.

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

[8] Zeyuan Allen-Zhu, Elad Hazan, Wei Hu and Yuanzhi Li. Linear convergence of a frank-wolfe type algorithm over trace norm balls. *NIPS*, pages 6192–6201, 2017.

[9] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. *Proceedings of the 30th International Conference on Machine Learning, ICML*, pages 427–435, 2013.

[10] Gauthier Gidel, Tony Jebara and Simon Lacoste-Julien. Frank-wolfe algorithms for saddle point problems. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, pages 362–371, 2017.

[11] Gauthier Gidel, Tony Jebara and Simon Lacoste-Julien. Frank-wolfe splitting via augmented lagrangian method. *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, pages 1456–1465, 2018.

[12] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[13] Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optomization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.

[14] Alp Yurtsever, Olivier Fercoq, Francesco Locatello and Volkan Cevher. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pages 5713–5722, 2018.

[15] Cun Mu, Yuqian Zhang, John Wright, and Donald Goldfarb. Scalable robust matrix recovery: Frank–wolfe meets proximal methods. *SIAM Journal on Scientific Computing*, 38(5):A3291–A3317, 2016.

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

[17] Dan Garber, Shoham Sabach and Atara Kaplan. Fast generalized conditional gradient method with applications to matrix recovery problems. *CoRR*, abs/1802.05581, 2018.

[18] Emile Richard, Pierre-Andr'e Savalle and Nicolas Vayatis. Estimation of simultaneously sparse and low rank matrices. *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[19] Yu-Xiang Wang, Huan Xu, and Chenlei Leng. Provable subspace clustering: When lrr meets ssc. In *Advances in Neural Information Processing Systems*, pages 64–72, 2013.

[20] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, pages 641–649, 2013.

[21] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.