Fast Stochastic Variance Reduced ADMM for Stochastic Composition Optimization

Yue Yu and **Longbo Huang**
Institute for Interdisciplinary Information Sciences, Tsinghua University
yu-y14@mails.tsinghua.edu.cn, longbohuang@tsinghua.edu.cn

Abstract

We consider the stochastic composition optimization problem proposed in [Wang *et al.*, 2016a], which has applications ranging from estimation to statistical and machine learning. We propose the first ADMM-based algorithm named com-SVR-ADMM, and show that com-SVR-ADMM converges linearly for strongly convex and Lipschitz smooth objectives, and has a convergence rate of $O(\log S/S)$, which improves upon the $O(S^{-4/9})$ rate in [Wang *et al.*, 2016b] when the objective is convex and Lipschitz smooth. Moreover, com-SVR-ADMM possesses a rate of $O(1/\sqrt{S})$ when the objective is convex but without Lipschitz smoothness. We also conduct experiments and show that it outperforms existing algorithms.

1 Introduction

Recently, [Wang *et al.*, 2016a] proposed the stochastic composition optimization of the following form:

$$\min_x (\mathbf{E}_i f_i \circ \mathbf{E}_j g_j)(x). \tag{1}$$

Here $x \in \mathbb{R}^q$, $\mathbf{E}_i f_i = \mathbf{E}_i\{f_i(x)\}$, $(f \circ g)(x) \triangleq f(g(x))$ denotes the composite function, and i, j are random variables. Problem (1) has been shown in [Wang *et al.*, 2016a] to include several important applications in estimation and machine learning.

In this paper, we focus on extending the formulation to include linear constraints, and consider the following variant of Problem (1):

$$\min_{x,\omega} \quad F(x) + R(\omega) \tag{2}$$
$$\text{s.t.} \quad Ax + B\omega = 0. \tag{3}$$

Here $F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(\frac{1}{m}\sum_{j=1}^{m} g_j(x))$, $x \in \mathbb{R}^q$, $\omega \in \mathbb{R}^l$, $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times l}$, $g_j : \mathbb{R}^q \mapsto \mathbb{R}^r$ and $f_i : \mathbb{R}^r \mapsto \mathbb{R}$ are continuous functions, and $R(\omega) : \mathbb{R}^l \mapsto \mathbb{R}$ is a closed convex function. The reason to consider the specific form of Problem (2) is as follows. (i) In practice, random variables such as i and j are obtained from problem-dependent data sets. Thus, they often only take values in a finite set with certain frequencies (captured by the first term in the objective (2)). (ii) Such problems often require the solutions to satisfy certain regularizing conditions (imposed by the term $R(\omega)$ and constraint (3)). Note here that the uniform distribution of i and j (the $\frac{1}{n}$ and $\frac{1}{m}$) in (2) is not critical. In Section 4, we show that our algorithm is also applicable under other distributions.

1.1 Motivating Examples

We first present a few motivating examples of formulation (2). The first example is a risk-averse learning problem discussed in [Wang *et al.*, 2016b], which can be formulated into the following mean-variance minimization problems, i.e.,

$$\min_x \quad \mathbf{E}_\epsilon h_\epsilon(x) + \lambda \mathbf{Var}_\epsilon h_\epsilon(x) \tag{4}$$
$$\text{s.t.} \quad Ax = 0. \tag{5}$$

Here $h_\epsilon(x)$ is the loss function w.r.t variable x and is parameterized by random variable ϵ, and $\mathbf{Var}_\epsilon(x) \triangleq \mathbf{E}_\epsilon\{[h_\epsilon(x) - \mathbf{E}_\epsilon h_\epsilon(x)]^2\}$ denotes its variance. We see that this example is of the form (2), where $\mathbf{E}_\epsilon h_\epsilon(x)$ plays the role of the regularizer and the variance term is the composition functions. There are many other problems that can be formulated into the mean-variance optimization as in (4), e.g., portfolio management [Alexander and Baptista, 2004].

The second motivating example is dynamic programming [Sutton and Barto, 1998; Dai *et al.*, 2016]. In this case, one can often approximate the value of each state by an inner product of a state feature ϕ_s and a target variable w. Then, the policy learning problem can be formulated into minimizing the Bellman residual as follows:

$$\min_w \sum_{s=1}^{S} (\langle \phi_s, w\rangle - \sum_{s'} P^\pi_{s,s'}(r_{s,s'} + \gamma\langle \phi_{s'}, w\rangle))^2 + R(w), \tag{6}$$

where $P^\pi_{s,s'}$ denotes the transition probabilities under a policy π, and γ denotes the discounting factor. Note that this problem also has the form of Problem (2).

The third example is multi-stage stochastic programming [Shapiro *et al.*, 2014]. For example, a two-stage optimization scenario often requires solving the following problem:

$$\min_x \mathbf{E}_v(\min_y \mathbf{E}_{u|v}(U(x, v, y, u))).$$

Here x, y are decision variables, v, u are the corresponding random variables, and the function U is the utility function.

In this case, the expectation $\mathbf{E}_{u|v}(\cdot)$ is the inner function and $\min_y(\cdot)$ is the outer function in Problem (2).

From these examples, we see that formulation (2) is general and includes important applications. Thus, it is important to develop fast and robust algorithms for solving (2).

1.2 Related Works

The stochastic composition optimization problem was first proposed in [Wang *et al.*, 2016a], where two solution algorithms, Basic SCGD and accelerated SCGD, were proposed. The algorithms were shown to achieve a sublinear convergence rate for convex and strongly convex cases, and were also shown to converge to a stationary point in the nonconvex case. Later, [Wang *et al.*, 2016b] proposed a proximal gradient algorithm called ASC-PG to improve the convergence rate when both inner and outer functions are smooth. However, the convergence rate is sublinear and their results do not include the regularizer when the objective functions are not strongly convex. In [Lian *et al.*, 2016], the authors solved the finite sample case of stochastic composition optimization and obtained two linear-convergent algorithms based on the stochastic variance reduction gradient technique (SVRG) proposed in [Johnson and Zhang, 2013]. However, the algorithms do not handle the regularizer either.

The ADMM algorithm, on the other hand, was first proposed in [Glowinski and Marroco, 1975; Gabay and Mercier, 1976] and later reviewed in [Boyd *et al.*, 2011]. Since then, several ADMM-based stochastic algorithms have been proposed, e.g., [Ouyang *et al.*, 2013; Suzuki and others, 2013; Wang and Banerjee, 2013]. However, these algorithms all possess sublinear convergence rates. Therefore, several recent works tried to combine the variance reduction scheme and ADMM to accelerate convergence. For instance, SVRG-ADMM was proposed in [Zheng and Kwok, 2016a]. It was shown that SVRG-ADMM converges linearly when the objective is strongly convex, and has a sublinear convergence rate in the general convex case. Another recent work [Zheng and Kwok, 2016b] further proved that SVRG-ADMM converges to a stationary point with a rate $O(\frac{1}{T})$ when the objective is nonconvex. In [Qian and Zhou, 2016], the authors used acceleration technique in [Allen-Zhu, 2016; Hien *et al.*, 2016] to further improve the convergence rate of SVRG-ADMM. However, all aforementioned variance-reduced ADMM algorithms cannot be directly applied to solving the stochastic composition optimization problem.

1.3 Contribution

In this paper, we propose an efficient algorithm called com-SVR-ADMM, which combines ideas of SVRG and ADMM, to solve stochastic composition optimization. Our algorithm is based on the SVRG-ADMM algorithm proposed in [Zheng and Kwok, 2016a], which does not apply to composite optimization problems. We consider three different objective functions in Problem (2), and show that our algorithm achieves the following performance.

- When $F(x)$ is strongly convex and Lipschitz smooth, and $R(\omega)$ is convex, our algorithm converges linearly. This convergence rate is comparable with those of com-SVRG-1 and com-SVRG-2 in [Lian *et al.*, 2016]. However, com-SVRG-1 and com-SVRG-2 do not take the commonly used regularization penalty into consideration. Experimental results also show that com-SVR-ADMM converges faster than com-SVRG-1 and com-SVRG-2.
- When $F(x)$ is convex and Lipschitz smooth, and $R(\omega)$ is convex, com-SVR-ADMM has a sublinear rate of $O(\frac{\log(S+1)}{S})$, where S is the number of outer iterations.[1] This result outperforms the $O(S^{-4/9})$ convergence rate of ASC-PG in [Wang *et al.*, 2016b].[2]
- When $F(x)$ and $R(\omega)$ are general convex functions (not necessarily Lipschitz smooth), com-SVR-ADMM achieves a rate of $O(\frac{1}{\sqrt{S}})$. To the best of our knowledge, this is the first convergence result for stochastic composite optimization with general convex problems without Lipschitz smoothness.

1.4 Notation

For vector x and a positive definite matrix G, the G-norm of vector x is defined as $||x||_G = \sqrt{x^T G x}$. For a matrix X, $||X||$ denotes its spectral norm, $\sigma_{max}(X), \sigma_{min}(X)$ denote its largest and smallest eigenvalue, respectively. $X^\dagger$ denotes the pseudoinverse of X. $\tilde{\nabla} R(\omega)$ denotes a noisy subgradient of nonsmooth $R(\omega)$. For a function $g(x) : \mathbb{R}^q \mapsto \mathbb{R}^r$, $\partial g(x) \in \mathbb{R}^{r\times q}$ denotes its Jacobian matrix. $\nabla f_{i_k}(g(x))$ denotes the gradient of $f_{i_k}(\cdot)$ at point $y = g(x)$.

2 Algorithm

Recall that the stochastic composition problem we want to solve has the following form:

$$\begin{aligned} \min_{x,\omega} \quad & F(x) + R(\omega) \\ \text{s.t.} \quad & Ax + B\omega = 0. \end{aligned}$$

where $F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(\frac{1}{m}\sum_{j=1}^{m} g_j(x))$. For clarity, we denote $F(x) = \frac{1}{n}\sum_{i=1}^{n} F_i(x)$, $F_i(x) = f_i(g(x))$, $g(x) = \mathbf{E}g(x) = \frac{1}{m}\sum_{j=1}^{m} g_j(x)$. Therefore, $\nabla F_i(x) = (\partial g(x))^T \nabla f_i(g(x))$.

Our proposed procedure adopts the ADMM scheme. At every iteration the primal variables (x, ω) are obtained by minimizing the following augmented Lagrangian equation parameterized with $\rho > 0$, i.e.,

$$\begin{aligned} & L_\rho(x, \omega, \lambda) \\ & = F(x) + R(\omega) + \langle\, \lambda, Ax + B\omega \rangle\ + \frac{\rho}{2}||Ax + B\omega||_2^2. \end{aligned}$$

The update of dual variable λ is similar to that under gradient descent with the stepsize equaling ρ. We also based our algorithm on a *sampling oracle* as in [Wang *et al.*, 2016b]. Specifically, we assume a stochastic first-order oracle, which,

[1] The number of inner iterations is constant.

[2] Note that ASC-PG is not based on SVRG and does not have inner loops.

Algorithm 1 com-SVR-ADMM for strongly convex stochastic composition optimization

1: **Input:** K, M, N, η, ρ, $\tilde{x}^0$, $\tilde{\omega}^0$, $\tilde{\lambda}^0 = -(A^T)^\dagger \nabla F(\tilde{x}^0)$;
2: **for** $s = 1, 2, ...$ **do**
3: $\tilde{x} = \tilde{x}^{s-1}$, $x^0 = \tilde{x}^{s-1}$, $\omega^0 = \tilde{\omega}^{s-1}$, $\lambda^0 = \tilde{\lambda}^{s-1}$;
4: $g(\tilde{x}) = \frac{1}{m}\sum_{j=1}^{m} g_j(\tilde{x})$; ($m$ queries)
5: evaluate $\nabla F(\tilde{x})$; ($m + n$ queries)
6: **for** $k = 0$ **to** $K - 1$ **do**
7: $\omega^{k+1} = \arg\min_\omega R(\omega) + \langle \lambda^k, B\omega \rangle + \frac{\rho}{2}||Ax^k + B\omega||_2^2$;
8: uniformly sample N_k and calculate $\hat{g}(x^k)$ using (9); ($2N$ queries)
9: uniformly sample i_k, j_k and calculate $\nabla \hat{F}_{i_k}(x^k)$ using (8); (4 queries)
10: $x^{k+1} = \arg\min_x \langle \nabla \hat{F}_{i_k}(x^k), x - x^k \rangle + \langle \lambda^k, Ax \rangle + \frac{\rho}{2}||Ax + B\omega^{k+1}||_2^2 + \frac{1}{2\eta}||x - x^k||_2^2$;
11: $\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + B\omega^{k+1})$;
12: **end for**
13: $\tilde{x}^s = \frac{1}{K}\sum_{k=1}^{K} x^k$, $\tilde{w}^s = \frac{1}{K}\sum_{k=1}^{K} w^k$, $\tilde{\lambda}^s = -(A^T)^\dagger \nabla F(\tilde{x}^s)$;
14: **end for**
15: **Output:** $\tilde{x}^s$, $\tilde{w}^s$.

if queried, returns a noisy gradient/subgradient or function value of $f_i(\cdot)$ and $g_j(\cdot)$ at a given point.

In the following sections, we introduce the stochastic variance reduced ADMM algorithm for solving stochastic compositional optimization (com-SVR-ADMM). We present com-SVR-ADMM in three different scenarios, i.e., strongly convex and Lipschitz smooth, general convex and Lipschitz smooth, and general convex. Algorithm 1 shows how com-SVR-ADMM is used in the strongly convex case, while Algorithm 2 is for the second and third cases.

2.1 Compositional Stochastic Variance Reduced ADMM for Strongly Convex Functions

As in SVRG, com-SVR-ADMM has K inner loops inside each outer iteration. At every outer iteration, we need to keep track of a reference point $\tilde{x}$ (Step 3 in Algorithm 1) for computing $g(\tilde{x})$ defined as

$$g(\tilde{x}) = \frac{1}{m}\sum_{j=1}^{m} g_j(\tilde{x}), \tag{7}$$

and evaluate $\partial g(\tilde{x})$, which is the Jacobian matrix of $g(x)$ at point $\tilde{x}$. The updates of ω^{k+1} and λ^{k+1} are the same as those in batch ADMM [Boyd *et al.*, 2011]. The main difference lies in the update for x^{k+1}, in that we use a stochastic sample i_k and replace $F_{i_k}(x)$ with its first-order approximation, and then approximate $\nabla F_{i_k}(x^k)$ by

$$\begin{aligned}\nabla \hat{F}_{i_k}(x^k) = {} & (\partial g_{j_k}(x^k))^T \nabla f_{i_k}(\hat{g}(x^k)) \\ & -(\partial g_{j_k}(\tilde{x}))^T \nabla f_{i_k}(g(\tilde{x})) + \nabla F(\tilde{x}).\end{aligned} \tag{8}$$

Here i_k, j_k are uniformly sampled from $\{1, 2, ..., n\}$ and $\{1, ..., m\}$, respectively. $\hat{g}(x^k)$ is an estimation of $g(x^k)$ defined as follows:

$$\hat{g}(x^k) = g(\tilde{x}) - \frac{1}{N}\sum_{1 \le j \le N} \left(g_{\mathbb{N}_k[j]}(\tilde{x}) - g_{\mathbb{N}_k[j]}(x^k) \right), \tag{9}$$

where $\mathbb{N}_k$ is a mini-batch and is obtained by uniformly and randomly sampling from $\{1, ..., m\}$ for N times (with replacement) and $\mathbb{N}_k[j]$ is the jth element of $\mathbb{N}_k$. In step 10 of Algorithm 1, we add a proximal term $\frac{1}{2\eta}||x - x^k||_2^2$ to control the distance between the next point x^{k+1} and the current point x^k. The parameter η is a constant and plays the role of stepsize as in [Ouyang *et al.*, 2013].

Note that our estimated gradient $\nabla \hat{F}_{i_k}(x^k)$ is biased due to the composition objective function and its form is the same as com-SVRG-1 in [Lian *et al.*, 2016]. However, we remark that our algorithm is not a trivial extension of com-SVRG-1 due to the existence of linear constraint and Lagrangian dual variable. Moreover, com-SVR-ADMM can handle regularization penalty while com-SVRG-1 cannot. Also, the update of $\tilde{\lambda}^s$ uses the pseudoinverse of A. In the common case when A is sparse, one can use the efficient Lanczos algorithm [Golub and Van Loan, 2012] to compute $A^\dagger$. Note that step 10 in Algorithm 1 often involves computing $A^T A$. The memory complexity for this step can be alleviated by algorithms proposed in recent works, e.g., [Zheng and Kwok, 2016a; Zhang *et al.*, 2011].

2.2 Compositional Stochastic Variance Reduced ADMM for General Convex Functions

In this section, we describe how com-SVR-ADMM handles general convex composition problems with Lipschitz smoothness. Without strong convexity, we need to make subtle changes. As shown in Algorithm 2, besides changes in variable initialization and output, another key difference is the approximation of $\nabla F_{i_k}(x)$, where we use $g(x^k)$ instead of $\hat{g}(x^k)$, i.e.,

$$\begin{aligned}\nabla \hat{F}_{i_k}(x^k) = {} & (\partial g_{j_k}(x^k))^T \nabla f_{i_k}(g(x^k)) \\ & -(\partial g_{j_k}(\tilde{x}))^T \nabla f_{i_k}(g(\tilde{x})) + \nabla F(\tilde{x}).\end{aligned} \tag{10}$$

Note that in the cases of interest (see Assumption 1 below), the approximated gradient $\nabla \hat{F}_{i_k}(x^k)$ is unbiased. The next change is the stepsize for updating x. In step 10 of Algorithm 2, we use a positive definite matrix G_k in the proximal term.[3] Therefore, the stepsize depends on two parameters: η_s and G_k, as shown in (11), where s and k are the iteration counters for outer and inner iteration, respectively. Here L_F is a parameter of Lipschitz smoothness and will be specified in our assumptions in next section.

$$\begin{aligned}&\eta_s = \frac{1}{(s+1)L_F}, \quad G_0 \succeq G_1 \succeq G_2 \succeq ... \succeq G_{K-1}, \\ &G_0 = \frac{1}{s}I, \quad G_{K-1} = \frac{1}{s+1}I, \quad G_K = \frac{1}{s+1}I.\end{aligned} \tag{11}$$

[3] The corresponding proximal term of Algorithm 1 can be viewed to have $G_k = I$.

Algorithm 2 com-SVR-ADMM for general convex stochastic composition optimization

Input: S, K, N, η_s, ρ, $\tilde{x}^0 = \hat{x}^0$, $\hat{\omega}^0$, $\hat{\lambda}^0$, $\hat{G}^0 = I$;
for $s = 1, 2, ..., S$ **do**
  $\tilde{x} = \tilde{x}^{s-1}$, $x^0 = \hat{x}^{s-1}$, $\omega^0 = \hat{\omega}^{s-1}$, $\lambda^0 = \hat{\lambda}^{s-1}$, $G_0 = \hat{G}^{s-1}$;
  $g(\tilde{x}) = \frac{1}{m}\sum_{j=1}^{m} g_j(\tilde{x})$; ($m$ queries)
  evaluate $\nabla F(\tilde{x})$; ($m + n$ queries)
  for $k = 0$ **to** $K - 1$ **do**
    $\omega^{k+1} = \arg\min_\omega R(\omega) + \langle \lambda^k, B\omega\rangle + \frac{\rho}{2}||Ax^k + B\omega||_2^2$;
    calculate $g(x^k) = \frac{1}{m}\sum_{j=1}^{m} g_j(x^k)$; ($m$ queries)
    uniformly sample i_k, j_k and calculate $\nabla \hat{F}_{i_k}(x^k)$ using (10); (4 queries)
    $x^{k+1} = \arg\min_x \langle \nabla \hat{F}_{i_k}(x^k), x - x^k\rangle + \langle \lambda^k, Ax\rangle + \frac{\rho}{2}||Ax + B\omega^{k+1}||_2^2 + \frac{1}{2\eta_s}||x - x^k||_{G_k}^2$;
    $\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + B\omega^{k+1})$;
  end for
  $\tilde{x}^s = \frac{1}{K}\sum_{k=1}^{K} x^k$, $\tilde{\omega}^s = \frac{1}{K}\sum_{k=1}^{K} \omega^k$, $\tilde{\lambda}^s = \frac{1}{K}\sum_{k=1}^{K} \lambda^k$, $\hat{x}^s = x^K$, $\hat{\omega}^s = \omega^K$, $\hat{\lambda}^s = \lambda^K$, $\hat{G}^s = G_K$;
end for
Output: $\bar{x} = \frac{1}{S}\sum_{s=1}^{S} \tilde{x}^s$, $\bar{\omega} = \frac{1}{S}\sum_{s=1}^{S} \tilde{\omega}^s$.

That is, G_k is nonincreasing for $k = 0, 1, ..., K$. Then, according to the definition of G-norm and (11), we have:

$$\frac{1}{2\eta_s}||x - x^k||_{G_k}^2 = \frac{1}{2\eta_{s,k}}||x - x^k||_2^2, \tag{12}$$

where $\eta_{s,k} = \frac{\eta_s}{\mathbf{co}(G_k)}$ and $\mathbf{co}(G_k) = a$ if $G_k = aI$, and a is a scalar. Therefore, $\eta_{s,k}$ serves as the stepsize [Ouyang *et al.*, 2013], and it can be verified that $\eta_{s,k}$ satisfies the following properties:

$$\eta_{s,0} = \frac{s}{(s+1)L_F},\ \eta_{s,K-1} = \frac{1}{L_F},\ \eta_{s,K} = \frac{1}{L_F},\quad \eta_{s,0} \le \eta_{s,1} \le ... \le \eta_{s,K-1}. \tag{13}$$

That is, $\eta_{s,k}$ changes from $\frac{s}{(s+1)L_F}$ to $\frac{1}{L_F}$ in stage s. Note that even though $\eta_{s,k}$ is not a constant, it still has a reasonable value and does not need to vanish. This feature is helpful for convergence acceleration.

2.3 General Convex Functions without Lipschitz Smoothness

In the previous two sections, we present the procedures of com-SVR-ADMM for the strongly convex and general convex settings, both under the Lipschitz smooth assumption of $F(x)$. In this section, we further investigate the case when the smooth condition is relaxed. We still use Algorithm 2, except that the values η_s and G_k are changed to

$$\eta_s = \frac{1}{s+1},\quad G_0 \succeq G_1 \succeq G_2 \succeq ... \succeq G_{K-1},$$
$$G_0 = \frac{1}{\sqrt{s}}I,\quad G_{K-1} = \frac{1}{\sqrt{s+1}}I,\quad G_K = \frac{1}{\sqrt{s+1}}I. \tag{14}$$

Therefore, using the same technique in (12), it can be verified that $\eta_{s,k}$ in this setting changes from $\frac{\sqrt{s}}{s+1}$ to $\frac{1}{\sqrt{s+1}}$ in stage s and decreases to zero. Note that in this case, the number of oracle calls at each step is the same as that in section 2.2.

Although the algorithm we proposed appears similar to the SVRG-ADMM algorithm in [Zheng and Kwok, 2016a], it is very different due to the composition nature of the objective function (which is not considered in SVRG-ADMM) and the stochastic variance reduced gradients in (8) and (10). These differences make it impossible to directly apply SVRG-ADMM and require a very different analysis for the new algorithm. Readers interested in the full proofs can refer to our technical report [Yu and Huang, 2017].

3 Theoretical Results

In this section, we analyze the convergence performance of com-SVR-ADMM under the three cases described in section 2. Below, we first state our assumptions. Note that the assumptions are not restrictive and are commonly made in the literature, e.g., [Wang *et al.*, 2016b; Ouyang *et al.*, 2013; Wang *et al.*, 2016a; Zheng and Kwok, 2016b].

Assumption 1. *(i) For each $i \in \{1, ..., n\}$, F_i is convex and continuously differentiable, $R(\omega)$ is convex (can be non-smooth). Moreover, there exists an optimal primal-dual solution $(x^*, \omega^*, \lambda^*)$ for Problem (2).*

(ii) The feasible set $\mathbb{X}$ for x is bounded and denote $D = \max_{x,y\in\mathbb{X}} ||x - y||$.

(iii) For randomly sampled $i_k \in \{1, ..., n\}$, $j_k \in \{1, ..., m\}$ and $\forall x$, we assume the following unbiased properties:

$$E((\partial g_{j_k}(x))^T \nabla f_{i_k}(g(x))) = \nabla F(x),$$
$$E(\partial g_{j_k}(x)) = \partial g(x),\quad E(\nabla F_{i_k}(x)) = \nabla F(x). \tag{15}$$

Assumption 2. *F is strongly convex with parameter $\mu_F > 0$, i.e., $\forall x$,*

$$F(x) - F(x^*) \ge \langle \nabla F(x^*), x - x^*\rangle + \frac{\mu_F}{2}||x - x^*||_2^2. \tag{16}$$

Assumption 3. *Matrix A has full row rank.*

Assumption 4. *There exists a positive constant L_F, such that $\forall i \in \{1, ..., n\}$, $\forall j \in \{1, ..., m\}$ and $\forall x, y$, we have*

$$||(\partial g_j(x))^T \nabla f_i(g(x)) - (\partial g_j(y))^T \nabla f_i(g(y))|| \le L_F||x - y||.$$

Assumption 5. *For each $i \in \{1, ..., n\}$, f_i is Lipschitz smooth with positive parameter L_f, that is, $\forall x, y$, we have*

$$||\nabla f_i(y) - \nabla f_i(x)|| \le L_f||y - x||. \tag{17}$$

Assumption 6. *For every $j \in \{1, ..., m\}$, $\partial g_j(x)$ is bounded, and for all x, y, $\exists\, C_G, L_G > 0$ that satisfy*

$$||g_j(x) - g_j(y)|| \le C_G||x - y||,\quad ||\partial g_j(x)|| \le C_G,$$
$$||g_j(x) - g_j(y)|| \le L_G||x - y||^2. \tag{18}$$

For clarity, we also use the following notations used in the theorems:

$$\begin{aligned} u &= \begin{bmatrix} x \\ \omega \end{bmatrix},\ u^k = \begin{bmatrix} x^k \\ \omega^k \end{bmatrix},\ \tilde{u}^s = \begin{bmatrix} \tilde{x}^s \\ \tilde{\omega}^s \end{bmatrix},\ \bar{u} = \begin{bmatrix} \bar{x} \\ \bar{\omega} \end{bmatrix}, \\ G(u) &= F(x) - F(x^*) - \langle \nabla F(x^*), x - x^* \rangle \\ &\quad + R(\omega) - R(\omega^*) - \langle \tilde{\nabla} R(\omega^*), \omega - \omega^* \rangle. \end{aligned} \tag{19}$$

It can be verified that $G(u)$ is always non-negative due to the convexity of $F(x)$ and $R(\omega)$. The following theorem and corollary show that Algorithm 1 has a linear convergence rate.

Proposition 1. *Under Assumption 4, we have $\forall i \in \{1, ..., n\}$ and $\forall x, y$:*

$$||\nabla F_i(x) - \nabla F_i(y)|| \leq L_F ||x - y||, \tag{20}$$

i.e., each F_i is Lipschitz smooth. Moreover, it implies $||\nabla F(x) - \nabla F(y)|| \leq L_F ||x - y||$.

Theorem 1. *Under Assumptions 1, 2, 3, 4, 5 and 6, if $0 < \eta \leq 1/L_F$, then under Algorithm 1,*

$$\gamma_1 \mathbf{E}[G(\tilde{u}^s)] \leq \gamma_2 G(\tilde{u}^{s-1}), \tag{21}$$

where (denote $\sigma(N) = \sqrt{1/N}$)

$$\begin{aligned} \gamma_1 =& (2\eta - \frac{32\eta^2 C_G^4 L_f^2}{\mu_F N} - \frac{48\eta^2 L_F^2 + 8\eta D C_G L_f L_G \sigma(N)}{\mu_F}) K, \\ \gamma_2 =& (K+1)(\frac{32\eta^2 C_G^4 L_f^2}{\mu_F N} + \frac{48\eta^2 L_F^2 + 8\eta D C_G L_f L_G \sigma(N)}{\mu_F}) \\ &+ \frac{2}{\mu_F} + \frac{2\eta\rho||A^T A||}{\mu_F} + \frac{2L_F \eta}{\rho \sigma_{min}(AA^T)}. \end{aligned}$$

Corollary 1. *Suppose the conditions in Theorem 1 hold. Then, there exist positive $\Theta(1)$ constants K (number of inner iterations) and N (mini-batch size) such that $\gamma_1, \gamma_2 > 0, \gamma = \gamma_2/\gamma_1 < 1$. Thus, Algorithm 1 converges linearly.*

From Corollary 1, if we want to achieve $\mathbf{E}[G(\tilde{u}^s)] \leq \epsilon$, $\forall \epsilon > 0$, the number of steps we need to take is roughly $s \geq \log(\frac{G(\tilde{u}^0)}{\epsilon}) / \log(\frac{1}{\gamma})$. In each iteration, we need $2m + n + K(2N + 4)$ oracle calls. Therefore, the overall query complexity is $O((m + n + KN) \log \frac{1}{\epsilon})$. For comparison, the query complexity is $O((m + n + \kappa^4) \log(1/\epsilon))$ for com-SVRG-1 and $O((m + n + \kappa^3) \log(1/\epsilon))$ for com-SVRG-2 [Lian *et al.*, 2016], where κ is a parameter related to condition number. We will see in simulations in section 4 that the overall query complexity of com-SVR-ADMM is lower than com-SVRG-1 and com-SVRG-2.

Now we prove the convergence property of com-SVR-ADMM under Assumptions 1 and 4.

Theorem 2. *Under Assumptions 1 and 4 , if η_s and G_k are chosen as in (11), under Algorithm 2,*

$$\begin{aligned} &\mathbf{E}(G(\bar{u}) + \Lambda ||A\bar{x} + B\bar{\omega}||) \\ \leq& \frac{4L_F D^2 \log(S+1)}{S} + \frac{L_F D^2 \log S}{2KS} \\ &+ \frac{L_F D^2 + \rho D^2 ||A^T A|| + \frac{2}{\rho}||\hat{\lambda}^0 - \lambda^*||_2^2 + \frac{2}{\rho}\Lambda^2}{2KS}, \end{aligned} \tag{22}$$

where $\Lambda > 0$.

From Theorem 2, we see that com-SVR-ADMM has an $O(\frac{\log(S+1)}{S})$ convergence rate under the general convex and Lipschitz smooth condition. It improves upon the convergence rate $O(S^{-4/9})$ in the recent work [Wang *et al.*, 2016b]. In Theorem 2, we consider both the convergence property of function value and feasibility violation. Since $G(u)$ and $||A\bar{x} + B\bar{\omega}||$ are both non-negative, each term has an $O(\frac{\log(S+1)}{S})$ convergence rate.

In the following theorem, we show that our algorithm exhibits $O(\frac{1}{\sqrt{S}})$ convergence rate for both the objective value and feasibility violation, when the objective is a general convex function.

Assumption 7. *The gradients/subgradients of all f_i, F_i, g_j and $R(\omega)$ are bounded and $||\nabla F_i(x)|| \leq C_F$, $C_F > 0$. Moreover, B is invertible and A, B are bounded.*

Theorem 3. *Under Assumptions 1 and 7, denote $\dot{x}^s = \frac{1}{K}\sum_{k=0}^{K-1} x^k$, $\tilde{z}^s = \begin{bmatrix} \dot{x}^s \\ \tilde{w}^s \end{bmatrix}$, $\bar{z} = \frac{1}{S}\sum_{s=1}^{S} \tilde{z}^s$. If η_s and G_k are chosen as in (14), there exists a positive $\Theta(1)$ constant ρ such that, under Algorithm 2,*

$$\begin{aligned} &\mathbf{E}(G(\bar{z}) + \Lambda ||A\bar{x} + B\bar{\omega}||) \\ \leq& \frac{C_1(C_4 + C_F)}{\sqrt{S}} + \frac{D^2}{K\sqrt{S}} + \frac{C_3 \log(S+1)}{S} \\ &+ \frac{D^2 + \rho||A^T A||D^2 + \frac{2}{\rho}||\hat{\lambda}^0 - \lambda^*||_2^2 + \frac{2}{\rho}\Lambda^2}{2KS}, \end{aligned} \tag{23}$$

where Λ, C_1, C_3, C_4 are positive constants.

The reason for the introduction of $\bar{z}$ is similar to the step taken in [Ouyang *et al.*, 2013], and is due to the lack of Lipschitz smooth property. This result implies an $O(\frac{1}{\sqrt{S}})$ convergence rate for both objective value and feasibility violation.

4 Experiments

In this section, we conduct experiments and compare com-SVR-ADMM to existing algorithms. We consider two experiment scenarios, i.e., the portfolio management scenario from [Lian *et al.*, 2016] and the reinforcement learning scenario from [Wang *et al.*, 2016b]. Since the objective functions in both scenarios are strongly convex and Lipshitz smooth, we only provide results for Algorithm 1.

4.1 Portfolio Management

Portfolio management is usually formulated as mean-variance minimization of the following form:

$$\min_x -\frac{1}{n}\sum_{i=1}^{n}\langle r_i, x\rangle + \frac{1}{n}\sum_{i=1}^{n}(\langle r_i, x\rangle - \frac{1}{n}\sum_{j=1}^{n}\langle r_j, x\rangle)^2 + R(x), \tag{24}$$

where $r_i \in \mathbb{R}^N$ for $i \in \{1, ..., n\}$, N is the number of assets, and n is the number of observed time slots. Thus, r_i is the observed reward in time slot i. We compare our proposed com-SVR-ADMM with three benchmarks: com-SVRG-1, com-SVRG-2 from [Lian *et al.*, 2016], and SGD. In order to compute the unbiased stochastic gradient of SGD, we first enumerate all samples in the data set of g to calculate $g(x)$ and

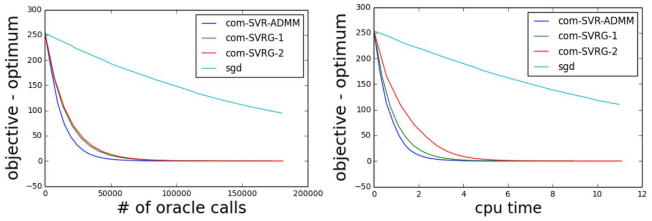

Figure 1: Portfolio Management with $cov = 2$.

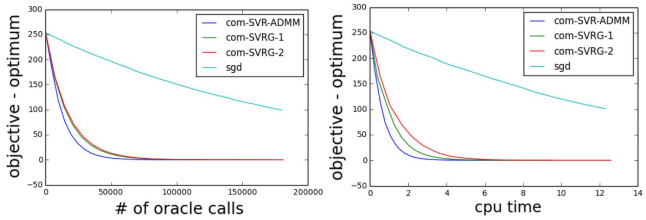

Figure 2: Portfolio Management with $cov = 10$. The other parameters have the same value as Figure 1.

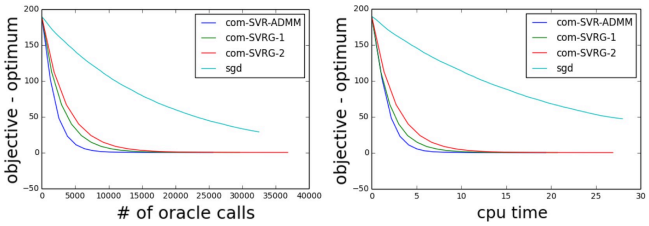

Figure 3: On-policy learning experiment with $S = 200$, $d = 100$.

$\partial g(x)$, then evaluate $\partial g(x)\nabla f_i(g(x))$ for a random sample i. Using the same definition of $g_j(x)$ and $f_i(y)$ and the same parameters generation method as [Lian *et al.*, 2016], we set the regularization to $R(x) = \frac{\mu}{2}||x||_2^2$, where $\mu > 0$.

The experimental results are shown in Figure 1 and Figure 2. Here the y-axis represents the objective value minus optimal value and the x-axis is the number of oracle calls or CPU time. We set $N = 200$, $n = 2000$. *cov* is the parameter used for reward covariance matrix generation [Lian *et al.*, 2016]. In Figure 1, $cov = 2$, and $cov = 10$ in Figure 2. All shared parameters in the four algorithms, e.g., stepsize, have the same values. We can see that all SVRG based algorithms perform much better than SGD, and com-SVR-ADMM outperforms two other linear convergent algorithms.

4.2 Reinforcement Learning

Here we consider the problem (6), which can be used for on-policy learning [Wang *et al.*, 2016b]. In our experiment, we assume there are finite states and the number of states is S. π is the policy in consideration. $P^{\pi}_{s,s'}$ is the transition probability from state s to s' given policy π, γ is a discount factor, $\phi_s \in \mathbb{R}^d$ is the feature of state s. Here we use a linear product $\langle \phi_s, w \rangle$ to approximate the value of state s. Our goal is to find the optimal $w \in \mathbb{R}^d$.

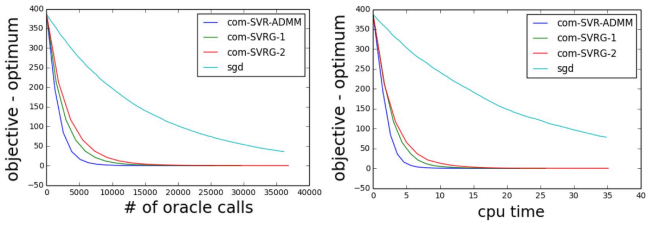

Figure 4: On-policy learning experiment with $S = 200$, $d = 200$. The other parameter values is the same as Figure 3.

We use the following specifications for oracles $g_{s'}(w)$ and $f_s(y)$:

$$g_{s'}(w) = (\phi_1^T w, r_{1,s'} + \gamma\phi_{s'}^T w, ..., \phi_S^T w, r_{S,s'} + \gamma\phi_{s'}^T w)^T,$$
$$f_s(y) = (y[2s-1] - y[2s])^2.$$

Note here $g_{s'}(w) \in \mathbb{R}^{2S}$, and $y[i]$ denote the i-th element of vector y. All shared parameters in four algorithms have the same values. Note here that the calculation of $\mathbf{E}[g(w)]$ is no longer under uniform distribution. We use the given transition probability. In this experiment, the transition probability is randomly generated and then regularized. The reward is also randomly generated. In addition, we include a regularization term $R(w) = \frac{\mu}{2}||w||_2^2$ with $\mu > 0$. The results are shown in Figure 3 and Figure 4. It can be seen that our proposed com-SVR-ADMM achieves faster convergence compared to the benchmark algorithms.

5 Conclusion

In this paper, we propose an ADMM-based algorithm, called com-SVR-ADMM, for stochastic composition optimization. We show that when the objective function is strongly convex and Lipschitz smooth, com-SVR-ADMM converges linearly. In the case when the objective function is convex (not necessarily strongly convex) and Lipschitz smooth, com-SVR-ADMM improves the theoretical convergence rate from $O(S^{-4/9})$ in [Wang *et al.*, 2016b] to $O(\frac{\log S}{S})$. When the objective is only assumed to be convex, com-SVR-ADMM has a convergence rate of $O(\frac{1}{\sqrt{S}})$. Experimental results show that com-SVR-ADMM outperforms existing algorithms.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China Grants 61672316, 61303195, the Tsinghua Initiative Research Grant, and the China youth 1000-talent grant.

References

[Alexander and Baptista, 2004] Gordon J Alexander and Alexandre M Baptista. A comparison of var and cvar constraints on portfolio selection with the mean-variance model. *Management science*, 50(9):1261–1273, 2004.

[Allen-Zhu, 2016] Zeyuan Allen-Zhu. Katyusha: Accelerated variance reduction for faster sgd. *ArXiv e-prints, abs/1603.05953*, 2016.

[Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[Dai *et al.*, 2016] Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from conditional distributions via dual kernel embeddings. *arXiv preprint arXiv:1607.04579*, 2016.

[Gabay and Mercier, 1976] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[Glowinski and Marroco, 1975] Roland Glowinski and A Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(2):41–76, 1975.

[Golub and Van Loan, 2012] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[Hien *et al.*, 2016] Le Thi Khanh Hien, Canyi Lu, Huan Xu, and Jiashi Feng. Accelerated stochastic mirror descent algorithms for composite non-strongly convex optimization. *arXiv preprint arXiv:1605.06892*, 2016.

[Johnson and Zhang, 2013] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[Lian *et al.*, 2016] Xiangru Lian, Mengdi Wang, and Ji Liu. Finite-sum composition optimization via variance reduced gradient descent. *arXiv preprint arXiv:1610.04674*, 2016.

[Ouyang *et al.*, 2013] Hua Ouyang, Niao He, Long Tran, and Alexander G Gray. Stochastic alternating direction method of multipliers. *ICML (1)*, 28:80–88, 2013.

[Qian and Zhou, 2016] Chao Zhang Hui Qian and Zebang Shen Tengfei Zhou. Accelerated stochastic admm for empirical risk minimization. *arXiv preprint arXiv:1611.04074*, 2016.

[Shapiro *et al.*, 2014] Alexander Shapiro, Darinka Dentcheva, et al. *Lectures on stochastic programming: modeling and theory*, volume 16. Siam, 2014.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Suzuki and others, 2013] Taiji Suzuki et al. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *ICML (1)*, pages 392–400, 2013.

[Wang and Banerjee, 2013] Huahua Wang and Arindam Banerjee. Online alternating direction method (longer version). *arXiv preprint arXiv:1306.3721*, 2013.

[Wang *et al.*, 2016a] Mengdi Wang, Ethan X Fang, and Han Liu. Stochastic compositional gradient descent: Algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2016.

[Wang *et al.*, 2016b] Mengdi Wang, Ji Liu, and Ethan X. Fang. Accelerating stochastic composition optimization. In *Advances In Neural Information Processing Systems*, pages 1714–1722, 2016.

[Yu and Huang, 2017] Yue Yu and Longbo Huang. Fast stochastic variance reduced admm for stochastic composition optimization. *arXiv preprint arXiv:1705.04138*, 2017.

[Zhang *et al.*, 2011] Xiaoqun Zhang, Martin Burger, and Stanley Osher. A unified primal-dual algorithm framework based on bregman iteration. *Journal of Scientific Computing*, 46(1):20–46, 2011.

[Zheng and Kwok, 2016a] Zheng and Kwok. Fast-and-light stochastic admm. *arXiv preprint arXiv:1604.07070*, 2016.

[Zheng and Kwok, 2016b] Shuai Zheng and James T Kwok. Stochastic variance-reduced admm. *arXiv preprint arXiv:1604.07070*, 2016.