Efficient Inexact Proximal Gradient Algorithm for Nonconvex Problems

Quanming Yao[1] **James T. Kwok**[1] **Fei Gao**[2] **Wei Chen**[2] **Tie-Yan Liu**[2]
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong[1]
Microsoft Research, Beijing, China[2]
{qyaoaa,jamesk}@cse.ust.hk, {feiga,wche,tyliu}@microsoft.com

Abstract

The proximal gradient algorithm has been popularly used for convex optimization. Recently, it has also been extended for nonconvex problems, and the current state-of-the-art is the nonmonotone accelerated proximal gradient algorithm. However, it typically requires two exact proximal steps in each iteration, and can be inefficient when the proximal step is expensive. In this paper, we propose an efficient proximal gradient algorithm that requires only one inexact (and thus less expensive) proximal step in each iteration. Convergence to a critical point is still guaranteed and has a $O(1/k)$ convergence rate, which is the best rate for nonconvex problems with first-order methods. Experiments on a number of problems demonstrate that the proposed algorithm has comparable performance as the state-of-the-art, but is much faster.

1 Introduction

In regularized risk minimization, we consider optimization problems of the form

$$\min_x F(x) \equiv f(x) + g(x), \tag{1}$$

where f is the loss, and g is the regularizer. Typically, f is smooth and convex (e.g., square and logistic losses), and g is convex but may not be differentiable (e.g., ℓ_1 and nuclear norm regularizers). The proximal gradient (PG) algorithm [Parikh and Boyd, 2014], together with its accelerated variant (APG) [Beck and Teboulle, 2009b; Nesterov, 2013], have been popularly used for solving this convex problem. Its crux is the proximal step $\text{prox}_g(\cdot) = \arg\min_x \frac{1}{2}\|x - \cdot\|_2^2 + \eta g(x)$, which can often be easily computed in closed-form.

While convex regularizers are easy to use, the resultant predictors may be biased [Zhang, 2010]. Recently, there is growing interest in the use of nonconvex regularizers, such as the log-sum-penalty [Candès *et al.*, 2008] and capped ℓ_1-norm [Zhang, 2010] regularizers. It has been shown that these often lead to sparser and more accurate models [Gong *et al.*, 2013; Lu *et al.*, 2014; Zhong and Kwok, 2014; Yao *et al.*, 2015]. However, the associated proximal steps become more difficult to compute analytically, and cheap closed-form solutions exist only for some simple nonconvex regularizers [Gong *et al.*, 2013]. This is further aggravated by the fact that the state-of-the-art PG algorithm for nonconvex optimization, namely the nonmonotone accelerated proximal gradient (nmAPG) algorithm [Li and Lin, 2015], needs more than one proximal steps in each iteration.

When the optimization objective is convex, one can reduce the computational complexity of the proximal step by only computing it inexactly (i.e., approximately). Significant speedup has been observed in practice, and the resultant inexact PG algorithm has the same convergence guarantee as the exact algorithm under mild conditions [Schmidt *et al.*, 2011]. However, on nonconvex problems, the use of inexact proximal steps has not been explored. Moreover, convergence of nmAPG hinges on the use of exact proximal steps.

In this paper, we propose a new PG algorithm for nonconvex problems. Unlike nmAPG, it performs only one proximal step in each iteration. Moreover, the proximal step can be inexact. The algorithm is guaranteed to converge to a critical point of the nonconvex objective. Experimental results on nonconvex total variation models and nonconvex low-rank matrix learning show that the proposed algorithm is much faster than nmAPG and other state-of-the-art, while still producing solutions of comparable quality.

The rest of the paper is organized as follows. Section 2 provides a brief review on the PG algorithm and its accelerated variant. The proposed algorithm is described in Section 3, and its convergence analysis studied in Section 4. Experimental results are presented in Section 5, and the last section gives some concluding remarks.

2 Related Work

In this paper, we assume that f in (1) is L-Lipschitz smooth (i.e., $\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2$), and g is proper, lower semi-continuous. Besides, $F = f + g$ in (1) is bounded from below, and $\lim_{\|x\|_2 \to \infty} F(x) = \infty$. Moreover, both f and g can be nonconvex.

First, we consider the case where f and g in (1) are convex. At iteration k, the accelerated proximal gradient (APG) algorithm generates x_{k+1} as

$$y_k = x_k + \theta_k(x_k - x_{k-1}), \tag{2}$$
$$x_{k+1} = \text{prox}_{\eta g}(y_k - \eta\nabla f(y_k)), \tag{3}$$

where $\theta_k = \frac{k-1}{k+2}$ and η is the stepsize [Beck and Teboulle, 2009b; Nesterov, 2013]. When $\theta_k = 0$, APG reduces to the plain PG algorithm.

On nonconvex problems, y_k can be a bad extrapolation and the iterations in (2), (3) may not converge [Beck and Teboulle, 2009a]. Recently, a number of PG extensions have been proposed to alleviate this problem. The iPiano [Ochs *et al.*, 2014], NIPS [Ghadimi and Lan, 2015], and UAG [Ghadimi and Lan, 2015] algorithms allow f to be nonconvex, but still requires g to be convex. The GD algorithm [Attouch *et al.*, 2013] also allows g to be nonconvex, but does not support acceleration.

The current state-of-the-art is the nonmonotone APG (nmAPG) algorithm [1] [Li and Lin, 2015], shown in Algorithm 1. It allows both f and g to be nonconvex, and also uses acceleration. To guarantee convergence, nmAPG ensures that the objective is sufficiently reduced in each iteration:

$$F(x_{k+1}) \leq F(x_k) - \frac{\delta}{2}\|v_{k+1} - x_k\|_2^2, \tag{4}$$

where $v_{k+1} = \text{prox}_{\eta g}(x_k - \eta\nabla f(x_k))$ and $\delta > 0$ is a constant. A second proximal step has to be performed (step 8) if a variant of (4) is not met (step 5).

Algorithm 1 Nonmonotone APG (nmAPG).

```
Require: choose $\eta \in (0, 1/L)$, a positive constant $\delta$, $\Delta_1 = F(x_1)$, $q_1 = 1$, and $\nu \in (0, 1)$;
 1: $x_0 = x_1 = x_1^a = 0$ and $t_0 = t_1 = 1$;
 2: for $k = 1, \dots, K$ do
 3:     $y_k = x_k + \frac{t_{k-1}}{t_k}(x_k^a - x_{k-1}) + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$;
 4:     $x_{k+1}^a = \text{prox}_{\eta g}(y_k - \eta\nabla f(y_k))$;
 5:     if $F(x_{k+1}^a) \leq \Delta_k - \frac{\delta}{2}\|x_{k+1}^a - y_k\|_2^2$ then
 6:         $x_{k+1} = x_{k+1}^a$;
 7:     else
 8:         $x_{k+1}^p = \text{prox}_{\eta g}(x_k - \eta\nabla f(x_k))$;
 9:         $x_{k+1} = \begin{cases} x_{k+1}^a & F(x_{k+1}^a) \leq F(x_{k+1}^p) \\ x_{k+1}^p & \text{otherwise} \end{cases}$;
10:     end if
11:     $q_{k+1} = \nu q_k + 1$;
12:     $t_{k+1} = \frac{1}{2}\left((4t_k^2 + 1)^{1/2} + 1\right)$;
13:     $\Delta_{k+1} = \frac{1}{q_{k+1}}(\nu q_k \Delta_k + F(x_{k+1}))$;
14: end for
15: return $x_{K+1}$.
```

3 Efficient APG for Nonconvex Problems

The proposed algorithm is shown in Algorithm 2. Following [Schmidt *et al.*, 2011], we use a simpler acceleration scheme in step 3. Efficiency of the algorithm comes from two key ideas: reducing the number of proximal steps to one in each iteration (Section 3.1); and the use of inexact proximal steps (Section 3.2). Besides, we also allow nonmonotone update on the objective (and so $F(y_k)$ may be larger than $F(x_k)$). This helps to jump from narrow curved valley and improve convergence [Grippo and Sciandrone, 2002; Wright *et al.*, 2009; Gong *et al.*, 2013]. Note that when the proximal step is inexact, nmAPG does not guarantee convergence as its Lemma 2 no longer holds.

[1]A less efficient monotone APG (mAPG) algorithm is also proposed in [Li and Lin, 2015].

Algorithm 2 Noconvex inexact APG (niAPG) algorithm.

```
Require: choose $\eta \in (0, \frac{1}{L})$ and $\delta \in (0, \frac{1}{\eta} - L)$;
 1: $x_0 = x_1 = 0$;
 2: for $k = 1, \dots, K$ do
 3:     $y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$;
 4:     $\Delta_k = \max_{t=\max(1,k-q),\dots,k} F(x_t)$;
 5:     if $F(y_k) \leq \Delta_k$ then
 6:         $v_k = y_k$;
 7:     else
 8:         $v_k = x_k$;
 9:     end if
10:     $z_k = v_k - \eta\nabla f(v_k)$;
11:     $x_{k+1} = \text{prox}_{\eta g}(z_k)$; // possibly inexact
12: end for
13: return $x_{K+1}$.
```

3.1 Using Only One Proximal Step

Recall that on extending APG to nonconvex problems, the key is to ensure a sufficient decrease of the objective in each iteration. Let $\rho = 1/\eta$. For the standard PG algorithm with exact proximal steps, the decrease in F can be bounded as follows.

Proposition 3.1 ([Gong *et al.*, 2013; Attouch *et al.*, 2013])**.** $F(x_{k+1}) \leq F(x_k) - \frac{\rho-L}{2}\|x_{k+1} - x_k\|_2^2$.

In nmAPG (Algorithm 1), there is always a sufficient decrease after performing the (non-accelerated) proximal descent from x_k to x_{k+1}^p (Proposition 3.1), but not necessarily the case for the accelerated descent from x_k to x_{k+1}^a (which is generated by a possibly bad extrapolation y_k). Hence, nmAPG needs to perform extra checking at step 5. If the condition fails, x_{k+1}^p is used instead of x_{k+1}^a in step 9.

As the main problem is on y_k, we propose to check $F(y_k)$ (step 5 in Algorithm 2) **before** the proximal step (step 11), instead of checking after the proximal steps. Though this change is simple, the main difficulty is how to guarantee convergence while simultaneously maintaining acceleration and using only one proximal step. As will be seen in Section 4.1, existing proofs do not hold even with exact proximal steps.

The following shows that a similar sufficient decrease condition can still be guaranteed after this modification.

Proposition 3.2. *With exact proximal steps in Algorithm 2,* $F(x_{k+1}) \leq \min(F(y_k), \Delta_k) - \frac{\rho-L}{2}\|x_{k+1} - v_k\|_2^2$.

In step 4, setting $q = 0$ is the most straightforward. The $F(y_k)$ value is then checked w.r.t. the most recent $F(x_k)$. The use of a larger q is inspired from the Barzilai-Borwein scheme for unconstrained smooth minimization [Grippo and Sciandrone, 2002]. This allows y_k to occasionally increase the objective, while ensuring $F(y_k)$ to be smaller than the largest objective value from the last q iterations. In the experiments, q is set to 5 as in [Wright *et al.*, 2009; Gong *et al.*, 2013].

3.2 Inexact Proximal Step

Proposition 3.2 requires exact proximal step, which can be expensive. Inexact proximal steps are much cheaper, but the inexactness has to be carefully controlled to ensure convergence. We will propose two such schemes depending on whether g is convex. Note that f is not required to be convex.

Convex g**.** As g is convex, the optimization problem associated with the proximal step is also convex. Let $h_{\eta g}(x) \equiv \frac{1}{2}\|x - z_k\|_2^2 + \eta g(x)$ be the objective in the proximal step. For any z_k, the dual of the proximal step at z_k can be obtained as

$$\max_w \mathcal{D}_{\eta g}(w) \equiv \eta\left(z_k^\top w - g^*(w)\right) - \frac{\eta^2}{2}\|w\|_2^2, \qquad (5)$$

where g^* is the convex conjugate of g. In an inexact proximal step, the obtained dual variable $\tilde{w}_k$ only approximately maximizes (5). The duality gap $\varepsilon_k \equiv h_{\eta g}(x_{k+1}) - \mathcal{D}_{\eta g}(\tilde{w}_k)$, where $x_{k+1} = z_k - \eta\tilde{w}_k$, upper-bounds the approximation error of the inexact proximal step $\epsilon_k \equiv h_{\eta g}(x_{k+1}) - h_{\eta g}\left(\text{prox}_{\eta g}(z_k)\right)$. To ensure the inexactness ϵ_k to be smaller than a given threshold τ_k, we can control the duality gap as $\varepsilon_k \le \tau_k$ [Schmidt *et al.*, 2011].

The following shows that x_{k+1} satisfies a similar sufficient decrease condition as in Proposition 3.2. Note that this cannot be derived from [Schmidt *et al.*, 2011], which relies on the convexity of f.

Proposition 3.3. $F(x_{k+1}) \le F(v_k) - \frac{\rho - L}{2}\|x_{k+1} - v_k\|_2^2 + \rho\varepsilon_k$.

Nonconvex g**.** When g is nonconvex, the GD algorithm [Attouch *et al.*, 2013] allows inexact proximal steps. However, it does not support acceleration, and nonmonotone update. Thus, its convergence proof cannot be used here.

As g is nonconvex, it is difficult to derive the dual of the corresponding proximal step, and the optimal duality gap may also be nonzero. Thus, we monitor the progress of F instead. Inspired by Proposition 3.1, we require x_{k+1} from an inexact proximal step to satisfy the following weaker condition:

$$F(x_{k+1}) \le F(v_k) - \frac{\delta}{2}\|x_{k+1} - v_k\|_2^2, \qquad (6)$$

where $\delta \in (0, \rho - L)$. This condition has also been used in the GD algorithm. However, it requires checking an extra condition which is impractical.[2]

We could have also used condition (6) when g is convex. However, Proposition 3.3 offers more precise control, as it can recover (6) by setting $\varepsilon_k = \frac{\rho - L - \delta}{2\rho}\|x_{k+1} - v_k\|_2^2$ (note that $\delta < \rho - L$). Besides, the duality gap ε_k is readily produced by primal-dual algorithms, and is often less expensive to compute than F.

[2]Specifically, the condition is: $\exists\ w_{k+1} \in \partial g(x_{k+1})$ such that $\|w_{k+1} + \nabla f(v_k)\|_2^2 \le b\|x_{k+1} - v_k\|_2^2$ for some constant $b > 0$. However, the subdifferential $\partial g(x_k)$ is in general difficult to compute.

4 Convergence Analysis

Definition 4.1 ([Attouch *et al.*, 2013])**.** *The* Frechet subdifferential *of F at x is*

$$\hat{\partial}F(x) = \left\{u : \lim_{y \ne x} \inf_{y \to x} \frac{F(y) - F(x) - u^\top(y - x)}{\|y - x\|_2} \ge 0\right\}.$$

The limiting subdifferential *(or simply* subdifferential*) of F at x is $\partial F(x) = \{u : \exists x_k \to x, F(x_k) \to F(x), u_k \in \hat{\partial}F(x_k) \to u,$ as $k \to \infty\}$.*

Definition 4.2 ([Attouch *et al.*, 2013])**.** *x is a* critical point *of F if $0 \in \nabla f(x) + \partial g(x)$.*

4.1 Exact Proximal Step

In this section, we will show that Algorithm 2 (where both f and g can be nonconvex) converges with a $O(1/K)$ rate. This is the best known rate for nonconvex problems with first-order methods [Nesterov, 2004]. A similar $O(1/K)$ rate for $\|\mathcal{G}(v_k)\|_2^2$ is recently established for APG with nonconvex f but only convex g [Ghadimi and Lan, 2015]. Note also that no convergence rate has been proved for nmAPG [Li and Lin, 2015] and GD [Attouch *et al.*, 2013] in this case. Besides, their proof techniques cannot be used here as their nonmonotone updates are different.

Theorem 4.1. *The sequence $\{x_k\}$ generated from Algorithm 2 (with exact proximal step) have at least one limit point, and all limit points are critical points of* (1).

Let $\mathcal{G}(v) = v - \text{prox}_{\eta g}(v - \eta\nabla f(v))$, the proximal mapping at v [Parikh and Boyd, 2014]. The following Lemma suggests that $\|\mathcal{G}(v)\|_2^2$ can be used to measure how far v is from optimality [Ghadimi and Lan, 2015].

Lemma 4.2 ([Gong *et al.*, 2013; Attouch *et al.*, 2013])**.** *v is a critical point of* (1) *if and only if $\mathcal{G}(v) = 0$.*

The following Proposition shows that the proposed Algorithm 2 converges with a $O(1/K)$ rate.

Proposition 4.3. *Let $\phi(k) = \arg\min_{t = \max(k-q,1),\dots,k} \|x_{t+1} - v_t\|_2^2$. (i) $\lim_{k\to\infty}\|\mathcal{G}(v_{\phi(k)})\|_2^2 = 0$; and (ii) $\min_{k=1,\dots,K}\|\mathcal{G}(v_{\phi(k)})\|_2^2 \le \frac{2(q+1)c_1}{(\rho - L)K}$, where $c_1 = \max_{t=1,\dots,q+1} F(x_t) - \inf F$.*

4.2 Inexact Proximal Step

Convex g**.** As in [Schmidt *et al.*, 2011], we assume that the duality gap ε_k decays as $O(1/k^{1+\varsigma})$ for some $\varsigma > 0$. Let $c \equiv \sum_{k=1}^{\infty}\varepsilon_k$. Note that $c < \infty$.

Theorem 4.4. *The sequence $\{x_k\}$ generated from Algorithm 2 have at least one limit point, and all limit points are critical points of* (1).

Proposition 4.5. *Let $e_k \equiv x_{k+1} - \text{prox}_{\eta g}(x_k - \eta\nabla f(x_k))$, the difference between the inexact and exact proximal step solutions at iteration k. We have $\|e_k\|_2^2 \le 2\varepsilon_k$.*

Note that the proof techniques in [Schmidt *et al.*, 2011] cannot be used here as f is not required to be convex. As in Proposition 4.3, we also use $\|\mathcal{G}(v_{\phi(k)})\|_2^2$ to measure how far $v_{\phi(k)}$ is from optimality.

Table 1: Results on the image inpainting experiment (CPU time is in seconds).

		$\lambda = 0.01$		$\lambda = 0.02$		$\lambda = 0.04$	
		RMSE	CPU time	RMSE	CPU time	RMSE	CPU time
(nonconvex)	GDPAN	0.0326±0.0001	212.1±50.9	0.0301±0.0001	172.6±28.4	0.0337±0.0001	151.6±57.0
	nmAPG	**0.0323±0.0001**	600.5±35.8	**0.0299±0.0001**	461.7±33.3	**0.0335±0.0001**	535.6±29.7
	niAPG(exact)	**0.0323±0.0001**	307.4±26.8	**0.0299±0.0001**	297.2±35.3	**0.0335±0.0001**	282.7±19.3
	niAPG	**0.0323±0.0002**	**91.6±10.8**	**0.0299±0.0001**	**77.1±7.4**	**0.0335±0.0001**	**56.5±9.4**
(convex)	ADMM	0.0377±0.0001	55.7±5.1	0.0337±0.0001	54.7±1.4	0.0362±0.0001	33.2±1.5

Table 2: Matrix completion performance on the synthetic data (CPU time in seconds). Here, NMSE is scaled by $\times 10^{-2}$. Group (I) is based on convex nuclear norm regularization; group (II) on factorization model; and group (III) on nonconvex model (8).

		$m = 500$ (observed: 12.43%)			$m = 1000$ (observed: 6.91%)			$m = 2000$ (observed: 3.80%)		
		NMSE	rank	CPU time	NMSE	rank	CPU time	NMSE	rank	CPU time
(I)	active	4.10±0.16	42	11.8±1.1	4.08±0.11	55	77.6±8.4	3.92±0.04	71	507.3±25.4
	ALT-Impute	3.99±0.15	42	1.9±0.2	3.87±0.09	55	29.4±1.2	3.68±0.03	71	143.1±3.9
(II)	AltGrad	2.99±0.45	5	0.2±0.1	2.73±0.21	5	**0.4±0.1**	2.67±0.27	5	**1.2±0.2**
	R1MP	23.04±1.27	45	0.3±0.1	21.39±0.94	54	0.9±0.1	20.11±0.28	71	2.7±0.2
(III)	IRNN	**1.96±0.05**	5	19.2±1.2	**1.88±0.04**	5	215.1±4.3	**1.80±0.03**	5	3009.5±35.9
	FaNCL	**1.96±0.05**	5	0.4±0.1	**1.88±0.04**	5	1.4±0.1	**1.80±0.03**	5	5.6±0.2
	nmAPG	**1.96±0.05**	5	2.3±0.2	**1.88±0.03**	5	6.9±0.3	**1.80±0.03**	5	27.1±4.0
	niAPG(exact)	**1.96±0.04**	5	1.8±0.2	**1.88±0.03**	5	5.3±0.5	**1.80±0.04**	5	18.4±2.2
	niAPG	**1.96±0.05**	5	**0.1±0.1**	**1.88±0.03**	5	**0.4±0.1**	**1.80±0.04**	5	**1.2±0.2**

Proposition 4.6. *(i)* $\lim_{k\to\infty} \|\mathcal{G}(v_{\phi(k)})\|_2^2 = 0$*; and (ii)* $\min_{k=1,\dots,K} \|\mathcal{G}(v_{\phi(k)})\|_2^2 \le \frac{2}{K}(4c + \frac{(q+1)(c_1+\rho c)}{\rho - L})$.

When all ε_k's are zero, Proposition 4.6 reduces to Proposition 4.3. In general, the bound of $\min_{k=1,\dots,K} \|\mathcal{G}(v_{\phi(k)})\|_2^2$ in Proposition 4.6 is larger due to the inexact proximal step.

Nonconvex g. With inexact proximal steps, nmAPG no longer guarantees convergence, and its proof cannot be easily extended. On the other hand, GD allows inexact proximal steps but uses a different approach to control inexactness. Moreover, it does not support acceleration.

The following shows that Algorithm 2 generates a bounded sequence, and Corollary 4.8 shows that the limit points are critical points.

Theorem 4.7. *The sequence $\{x_k\}$ generated from Algorithm 2 has at least one limit point.*

Corollary 4.8. *Let $\{x_{k_j}\}$ be a subsequence of $\{x_k\}$ with $\lim_{k_j\to\infty} x_{k_j} = x_*$. If (i) $x_{k+1} \ne v_k$ unless $v_k = \text{prox}_{\eta g}(v_k - \eta\nabla f(v_k))$, and (ii) $\lim_{k_j\to\infty} F(x_{k_j}) = F(x_*)$, then x_* is a critical point of* (1).

Assumption (i), together with Lemma 4.2, ensures that the sufficient decrease condition in (6) will not be trivially satisfied by $x_{k+1} = v_k$, unless v_k is a critical point. Assumption (ii) follows from Definition 4.1, as the subdifferential is defined by a limiting process.

Proposition 4.9. *[Attouch* et al.*, 2013] Assumption (ii) is satisfied when (i) the proximal step is exact; or (ii) g is continuous or is the indicator function of a compact set.*

When the proximal step is exact or when g is convex, $\mathcal{G}(\cdot)$ has been used to measure the distance from optimality. However, this is inappropriate when g is nonconvex and the proximal step is inexact, as the inexactness can no longer be directly controlled. Instead, we will measure optimality via $a_k \equiv \|x_{k+1} - v_k\|_2^2$.

Proposition 4.10. *(i)* $\lim_{k\to\infty} a_k = 0$*; and (ii)* $\min_{k=1,\dots,K} a_{\phi(k)} \le \frac{2(q+1)c_1}{\delta K}$.

When the proximal step is exact, $x_{k+1} = \text{prox}_{\eta g}(v_k - \eta\nabla f(v_k))$, and $a_k = \|\mathcal{G}(v_k)\|_2^2$. Proposition 4.10 then reduces to Proposition 4.3 (but with a looser bound).

5 Experiments

In this section, we perform experiments when g is convex (Section 5.1) and nonconvex (Section 5.2).

5.1 Image Inpainting

The total variation (TV) model [Beck and Teboulle, 2009a] has been popularly used in image processing. Let $y \in \mathbb{R}^d$ be the vectorized input image and $x \in \mathbb{R}^d$ be the recovered one. We consider the TV model with nonconvex log-sum-penalty regularizer [Candès *et al.*, 2008].

$$\min_x \frac{1}{2}\|M \odot (x - y)\|_2^2 + \lambda \sum_{i=1}^{d} \kappa([D_h x]_i) + \kappa([D_v x]_i), \quad (7)$$

where $M \in \{0,1\}^d$ is a mask such that $M_{ij} = 1$ indicates that the corresponding pixel is observed, D_h and D_v are the horizontal and vertical partial derivative operators, $\odot$ is the elementwise multiplication, and $\kappa(\alpha) = \log(1 + |\alpha|)$.

As suggested in [Yao and Kwok, 2016], (7) can be transformed as the minimization of $f(x) + \lambda\,\text{TV}(x)$, where $f(x) = \frac{1}{2}\|M \odot (x-y)\|_2^2 - \lambda[\text{TV}(x) + \sum_{i=1}^d \kappa([D_h x]_i) + \kappa([D_v x]_i)]$ is nonconvex but smooth, and $\text{TV}(x) = \|D_h x\|_1 + \|D_v x\|_1$ is the standard (convex) TV regularizer. Thus, we only need to handle the proximal step of the TV regularizer, which will be computed numerically by solving its dual using L-BFGS [Beck and Teboulle, 2009a].

The following solvers on the transformed problems are compared: (i) GDPAN [Zhong and Kwok, 2014], which performs gradient descent with the proximal average; (ii) nmAPG; (iii) the proposed niAPG, in which inexactness of the proximal step is controlled by decaying the duality gap ε_k at a rate of $O(1/k^{1.5})$; and (iv) the exact version of

Table 3: Results on the *MovieLens* data sets (CPU time in seconds). Here, RMSE is scaled by 10^{-1}. Group (I) is based on convex nuclear norm regularization; group (II) on factorization model; and group (III) on nonconvex model (8).

		MovieLens-1M			*MovieLens-10M*			*MovieLens-20M*		
		RMSE	rank	CPU time	RMSE	rank	CPU time	RMSE	rank	CPU time
(I)	active	8.20±0.01	68	50.5±1.6	8.14±0.01	101	1520.8±18.2	8.02±0.01	197	7841.9±666.3
	ALT-Impute	8.18±0.01	68	34.0±1.1	8.14±0.01	101	821.7±34.5	8.01±0.01	197	3393.2±220.3
(II)	AltGrad	8.02±0.03	6	4.0±1.1	7.97±0.04	9	94.5±30.8	7.94±0.04	10	298.3±54.1
	R1MP	8.53±0.02	13	**1.3±0.2**	8.52±0.04	23	**58.8±11.0**	8.54±0.02	26	**139.2±23.7**
(III)	FaNCL	**7.88±0.01**	5	12.5±0.9	**7.79±0.01**	8	703.5±18.3	**7.84±0.03**	9	2296.9±176.4
	nmAPG	**7.87±0.01**	5	12.5±0.9	**7.80±0.01**	8	627.5±16.4	**7.85±0.01**	9	1577.9±103.2
	niAPG(exact)	**7.87±0.01**	5	11.1±0.8	**7.79±0.01**	8	403.1±19.6	**7.84±0.01**	9	1111.9±65.3
	niAPG	**7.87±0.01**	5	2.7±0.3	**7.79±0.01**	8	90.2±2.6	**7.85±0.01**	9	257.6±33.4

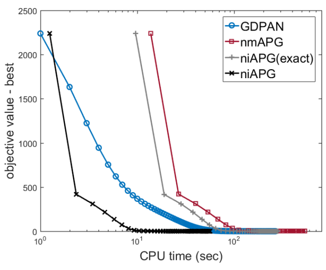
Table 4: Number of proximal steps on the synthetic and recommender system data sets.

	Synthetic			*MovieLens*			*Netflix*	*Yahoo*
	m =500	m =1000	m =2000	*1M*	*10M*	*20M*		
nmAPG	77	104	145	95	221	236	183	579
niAPG(exact)	64 (↓17%)	85 (↓18%)	115 (↓21%)	82 (↓13%)	143 (↓35%)	165 (↓31%)	133 (↓27%)	425 (↓26%)
niAPG	64 (↓17%)	85 (↓18%)	115 (↓21%)	81 (↓15%)	140 (↓36%)	160 (↓32%)	132 (↓28%)	413 (↓29%)

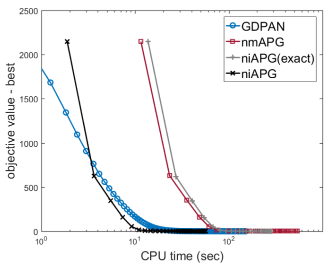
niAPG(exact), which simulates an exact proximal step with a small duality gap (10^{-4}). We do not compare with the GD algorithm [Attouch *et al.*, 2013], as its inexactness condition is difficult to check and it does not use acceleration.

As a further baseline, we compare with the convex TV model: $\min_x \frac{1}{2}\|M \odot (x-y)\|_2^2 + \lambda(\|D_v x\|_1 + \|D_h x\|_1)$, which is solved using ADMM [Boyd *et al.*, 2011]. We do not compare with CCCP [Yuille and Rangarajan, 2002], which is slow in practice [Yao and Kwok, 2016].

Experiments are performed on the "Lena" image. We normalize the pixel values to $[0, 1]$, and then add Gaussian noise from $\mathcal{N}(0, 0.05)$. 50% of the pixels are randomly sampled as observed. For performance evaluation, we report the CPU time and root-mean-squared error (RMSE) on the whole image. The experiment is repeated five times. Results are shown in Table 1.[3] Figure 1 plots convergence of the objective.[4] As can be seen, the nonconvex TV model has better RMSE than the convex one. Among the nonconvex models, niAPG is much faster, as it only requires a small number of cheap inexact proximal steps (Table 5).



(a) $\lambda = 0.02$.



(b) $\lambda = 0.04$.

Figure 1: Objective value vs CPU time (in seconds) on the image data.

[3]In all the tables, the boldface indicates the best and comparable results (according to the pairwise t-test with 95% confidence).

[4]Because of the lack of space, the plot for $\lambda = 0.01$ is not shown.

Table 5: Number of proximal steps on image data. Number in brackets is the percentage reduction w.r.t. nmAPG.

	$\lambda = 0.01$	$\lambda = 0.02$	$\lambda = 0.04$
nmAPG	87	46	43
niAPG(exact)	47 (↓46%)	35 (↓24%)	29 (↓33%)
niAPG	57 (↓35%)	41 (↓11%)	28 (↓35%)

5.2 Matrix Completion

In this section, we consider matrix completion with a nonconvex low-rank regularizer. As shown in [Lu *et al.*, 2014; Yao *et al.*, 2015; Yao and Kwok, 2015], it gives better performance than nuclear-norm based and factorization approaches. The optimization problem can be formulated as

$$\min_{\text{rank}(X) \le r} \frac{1}{2}\|P_\Omega(X_{ij} - O_{ij})\|_F^2 + \lambda \sum_{i=1}^{r} \kappa\left(\sigma_i(X)\right), \quad (8)$$

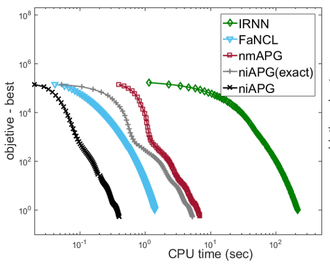
where O_{ij}s are the observations, $\Omega_{ij} = 1$ if $O_{i,j}$ is observed, and 0 otherwise, $\sigma_i(X)$ is the ith leading singular value of X, and r is the desired rank. The associated proximal step can be solved with rank-r SVD [Lu *et al.*, 2014].

Synthetic Data. The observed $m \times m$ matrix is generated as $O = UV + G$, where the entries of $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{k \times m}$ (with $k = 5$) are sampled i.i.d. from the normal distribution $\mathcal{N}(0, 1)$, and entries of G sampled from $\mathcal{N}(0, 0.1)$. A total of $\|\Omega\|_1 = 2mk\log(m)$ random entries in O are observed. Half of them are used for training, and the rest as validation set.

In the proposed niAPG algorithm, its proximal step is approximated by using power method [Halko *et al.*, 2011], and inexactness of the proximal step is monitored by condition (6). Its variant niAPG(exact) has exact proximal steps computed by the Lancoz algorithm [Larsen, 1998]. They are compared with the following solvers on the nonconvex model (8): (i) Iterative reweighted nuclear norm (IRNN) [Lu *et al.*, 2014] algorithm; (ii) Fast nonconvex low-rank learning (FaNCL) algorithm [Yao *et al.*, 2015], using the power method to approximate the proximal step; (iii) nmAPG, in which the proximal step is exactly computed by the Lancoz algorithm. We also compare with other matrix completion algorithms, including the well-known (convex) nuclear-norm-regularized algorithms: (i) active subspace selection [Hsieh and Olsen, 2014] and (ii) ALT-Impute [Hastie *et al.*, 2015]. We also compare with state-of-the-art factorization models (where the rank is tuned by the validation set): (i) R1MP [Wang *et al.*, 2015]; and (ii) state-of-the-art gradient descent based AltGrad [Zhao *et al.*, 2015]. We do not compare with the Frank-Wolfe algorithm [Zhang *et al.*, 2012], which has been shown to be slower [Hsieh and Olsen, 2014].

Testing is performed on the non-observed entries (denoted $\bar{\Omega}$). Three measures are used for performance evaluation: (i) normalized mean squared error (NMSE): $\sqrt{\|P_{\bar{\Omega}}(X - UV)\|_F^2}/\sqrt{\|P_{\bar{\Omega}}(UV)\|_F^2}$; (ii) rank of X; and (iii) training time. Each experiment is repeated five times.

Table 2 shows the performance. Convergence for algorithms solving (8) is shown[5] in Figure 2. As has also been observed in [Lu *et al.*, 2014; Yao *et al.*, 2015], nonconvex regularization yields lower NMSE than nuclear-norm-regularized and factorization models. Again, niAPG is the fastest. Its speed is comparable with AltGrad, but is more accurate. Table 4 compares the numbers of proximal steps. As can be seen, both nmAPG(exact) and nmAPG require significantly fewer proximal steps.


(a) $m = 1000$.

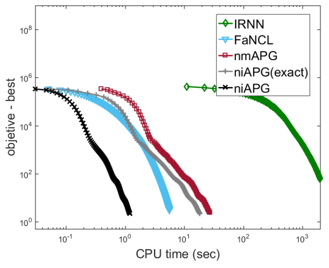

(b) $m = 2000$.

Figure 2: Objective value vs CPU time (in seconds) on the synthetic matrix completion data set.

Recommender Systems. We first consider the *MovieLens* data sets (Table 6), which contain ratings of different users on movies or musics. We follow the setup in [Wang *et al.*, 2015; Yao *et al.*, 2015; Yao and Kwok, 2015], and use 50% of the observed ratings for training, 25% for validation and the rest for testing. For performance evaluation, we use the root mean squared error on the test set $\bar{\Omega}$: RMSE $= \sqrt{\|P_{\bar{\Omega}}(O - X)\|_F^2}/\sqrt{\|\bar{\Omega}\|_1}$, rank of the recovered matrix X, and CPU time. The experiment is repeated five times.

Table 6: Recommender system data sets used.

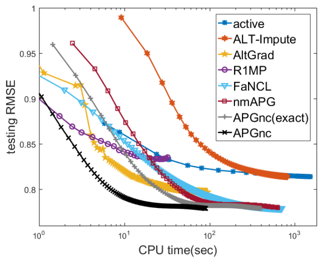
		#users	#items	#ratings
MovieLens	*1M*	6,040	3,449	999,714
	10M	69,878	10,677	10,000,054
	20M	138,493	26,744	20,000,263
Netflix		480,189	17,770	100,480,507
Yahoo		1,000,990	624,961	262,810,175

Table 3 shows the recovery performance. IRNN is not compared as it is too slow. Again, the nonconvex model consistently outperforms nuclear-norm-regularized and factorization models. R1MP is the fastest, but its recovery performance is poor. Figure 3(a) shows the convergence, and Table 4 compares the numbers of proximal steps. niAPG(exact) is faster than nmAPG due to the use of fewer proximal steps. niAPG is even faster with the use of inexact proximal steps.

Finally, we perform experiments on the large *Netflix* and *Yahoo* data sets (Table 6). We randomly use 50% of the observed ratings for training, 25% for validation and the rest for testing. Each experiment is repeated five times. We do not compare with nuclear-norm-regularized methods as they yield higher rank and RMSE than others. Table 7 shows the recovery performance, and Figures 3(b) show the convergence. Again, niAPG is the fastest and most accurate.

Table 7: Results on the *Netfix* and *Yahoo* data sets. Here, RMSE is scaled by $\times 10^{-1}$.

		RMSE	rank	CPU time (min)
Netflix	AltGrad	8.16±0.02	15	221.7±5.6
	FaNCL	7.94±0.01	13	240.8±22.7
	nmAPG	**7.92±0.01**	13	132.8±2.1
	niAPG(exact)	**7.92±0.01**	13	97.7±1.8
	niAPG	**7.92±0.01**	13	**25.2±0.6**
Yahoo	AltGrad	6.69±0.01	14	112.9±4.2
	FaNCL	**6.54±0.01**	9	487.6±32.0
	nmAPG	**6.53±0.01**	9	184.3±6.3
	niAPG(exact)	**6.53±0.01**	9	140.7±5.8
	niAPG	**6.53±0.01**	9	**38.7±2.3**


(a) *MovieLens-10M*.

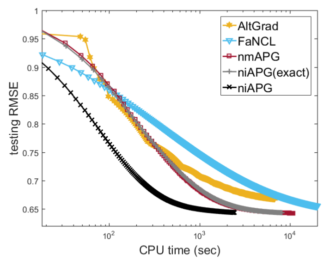

(b) *Yahoo*.

Figure 3: Testing RMSE vs CPU time (in seconds) on the recommendation system data sets.

6 Conclusion

In this paper, we proposed an efficient accelerated proximal gradient algorithm for nonconvex problems. Compared with the state-of-the-art [Li and Lin, 2015], the proximal step can be inexact and the number of proximal steps required is significantly reduced, while still ensuring convergence to a critical point. Experiments on image denoising and matrix completion problems show that the proposed algorithm has comparable (or even better) prediction performance as the state-of-the-art, but is much faster.

Acknowledgments

This research project is partially funded by Microsoft Research Asia and the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614513). The first author would thank helpful discussion and suggestions from Lu Hou and Yue Wang.

[5]Because of the lack of space, the plot for $m = 500$ is not shown.

References

[Attouch *et al.*, 2013] H. Attouch, J. Bolte, and B.F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[Beck and Teboulle, 2009a] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[Beck and Teboulle, 2009b] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[Boyd *et al.*, 2011] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[Candès *et al.*, 2008] E.J. Candès, M.B. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[Ghadimi and Lan, 2015] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):1–41, 2015.

[Gong *et al.*, 2013] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *ICML*, pages 37–45, 2013.

[Grippo and Sciandrone, 2002] L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Computational Optimization and Applications*, 23(2):143–169, 2002.

[Halko *et al.*, 2011] N. Halko, P.G. Martinsson, and J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[Hastie *et al.*, 2015] T. Hastie, R. Mazumder, J.D. Lee, and R. Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *Journal of Machine Learning Research*, 16:3367–3402, 2015.

[Hsieh and Olsen, 2014] C.J. Hsieh and P. Olsen. Nuclear norm minimization via active subspace selection. In *ICML*, pages 575–583, 2014.

[Larsen, 1998] R.M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. DAIMI PB-357, Department of Computer Science, Aarhus University, 1998.

[Li and Lin, 2015] H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *NIPS*, pages 379–387, 2015.

[Lu *et al.*, 2014] C. Lu, J. Tang, S Yan, and Z Lin. Generalized nonconvex nonsmooth low-rank minimization. In *CVPR*, pages 4130–4137, 2014.

[Nesterov, 2004] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[Nesterov, 2013] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[Ochs *et al.*, 2014] P. Ochs, Y. Chen, T. Brox, and T. Pock. iPiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.

[Parikh and Boyd, 2014] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[Schmidt *et al.*, 2011] M. Schmidt, N.L. Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *NIPS*, pages 1458–1466, 2011.

[Wang *et al.*, 2015] Z. Wang, M.J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye. Orthogonal rank-one matrix pursuit for low rank matrix completion. *SIAM Journal on Scientific Computing*, 37(1), 2015.

[Wright *et al.*, 2009] S.J. Wright, R.D. Nowak, and M.A. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[Yao and Kwok, 2015] Q. Yao and J.T. Kwok. Accelerated inexact Soft-Impute for fast large-scale matrix completion. In *The 24th International Conference on Artificial Intelligence*, pages 4002–4008, 2015.

[Yao and Kwok, 2016] Q. Yao and J.T. Kwok. Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity. In *ICML*, pages 2645–2654, 2016.

[Yao *et al.*, 2015] Q. Yao, J.T. Kwok, and W. Zhong. Fast low-rank matrix learning with nonconvex regularization. In *ICDM*, pages 539–548, 2015.

[Yuille and Rangarajan, 2002] A.L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In *NIPS*, pages 1033–1040, 2002.

[Zhang *et al.*, 2012] X. Zhang, D. Schuurmans, and Y.-L. Yu. Accelerated training for matrix-norm regularization: a boosting approach. In *NIPS*, pages 2906–2914, 2012.

[Zhang, 2010] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010.

[Zhao *et al.*, 2015] T. Zhao, Z. Wang, and H. Liu. A nonconvex optimization framework for low rank matrix estimation. In *NIPS*, pages 559–567, 2015.

[Zhong and Kwok, 2014] L.W. Zhong and J.T. Kwok. Gradient descent with proximal average for nonconvex and composite regularization. In *AAAI*, pages 2206–2212, 2014.