The Catalyst Acceleration Algorithm Report

By Xiangnan YUE

1 Introduction

This paper's background idea is based on extrapolation in Nesterov [17] and a new proximal point algorithm in Güler [9]. Nesterov's idea of controlling approximation error by using approximate estimate sequence provided the base for this paper's theorems' proofs and Güler's strategy for constructing the proximal point algorithm provided the tools for constructing this paper's Catalyst algorithm.

We will first present the main algorithm and ideas of the papers, and then give a brief description of main elements of the proofs, finally we present the results for a simple simulation showing the benefits of the Catalyst Acceleration.

2 The algorithm

The catalyst acceleration is a general method to wrap an algorithm $\mathscr{M}$ into an accelerated algorithm $\mathscr{A}$. This approach can accelerate a wide range of first-order optimization algorithms, from classical gradient descent to randomized ones like SAG, SAGA, SDCA, SVRG and Finito/MISO.

The minimization problem can be written as:

$$\min_{x \in R^p} \{F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \Phi(x)\} \qquad (1)$$

This general approach is listed as follows. This algorithm replaced the objective function F by an auxiliary $G_k(x)$ and needs to solve for the minimum of G for each round using an algorithm $\mathscr{M}$, which is to be accelerated. This step is the so-called inner loop or subproblem (5). We also note that (5) is actually a proximal operator on function F, and it can be non-differentiable at some point (eg. the lasso, or elastic-net). One issue is that G_k^* is not known for each round but we can use the duality gap as an upper bound of this approximation error $G_k(x_k) - G^*$.

Algorithm 1 Catalyst

---

input initial estimate $x_0 \in \mathbb{R}^p$, parameters κ and α_0, sequence $(\varepsilon_k)_{k \geq 0}$, optimization method $\mathcal{M}$;

1: Initialize $q = \mu/(\mu + \kappa)$ and $y_0 = x_0$;

2: **while** the desired stopping criterion is not satisfied **do**

3: Find an approximate solution of the following problem using $\mathcal{M}$

$$x_k \approx \arg\min_{x \in \mathbb{R}^p} \left\{ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_{k-1}\|^2 \right\} \quad \text{such that} \quad G_k(x_k) - G_k^* \leq \varepsilon_k. \quad (5)$$

4: Compute $\alpha_k \in (0, 1)$ from equation $\alpha_k^2 = (1 - \alpha_k)\alpha_{k-1}^2 + q\alpha_k$;

5: Compute

$$y_k = x_k + \beta_k (x_k - x_{k-1}) \quad \text{with} \quad \beta_k = \frac{\alpha_{k-1}(1 - \alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k}. \qquad (6)$$

6: **end while**

output x_k (final estimate).

---

3 Convergence Analysis

The analysis for the complexity of this algorithm always consists of two parts: the convergence rate for global procedures regardless of the subproblem (5), and the complexity of solving the approximation of minimization problem (5). The proof of the main theorems and propositions used Nesterov's Approximate Estimate Sequence, which is to construct an upper bound for the objective function's distance to the optimal. We state only the main Theorems and Propositions in this report.

3.1 Analysis of Strongly Convex Case

Theorem 3.1 (Convergence of Algorithm 1, μ-Strongly Convex Case).
Choose $\alpha_0 = \sqrt{q}$ *with* $q = \mu/(\mu + \kappa)$ *and*

$$\varepsilon_k = \frac{2}{9}(F(x_0) - F^*)(1-\rho)^k \quad \text{with} \quad \rho < \sqrt{q}.$$

Then, Algorithm 1 generates iterates $(x_k)_{k\geq 0}$ *such that*

$$F(x_k) - F^* \leq C(1-\rho)^{k+1}(F(x_0) - F^*) \quad \text{with} \quad C = \frac{8}{(\sqrt{q}-\rho)^2}. \tag{7}$$

Theorem 3.1 states the linear convergence rate of our algorithm. Combined with Proposition 3.2 which states that $G_k(z_t) - G_k^* \leq A(1-\tau_M)^t(G_k(z_0) - G_k^*)$ and by taking $z_0 = x_{k-1}$, the approximated precision demanded in the algorithm can be reached in $T_{\mathscr{M}} = \tilde{O}(1/\tau_M)$ iterations. Thus for $s = kT_{\mathscr{M}}$ iterations we got

$$F_s - F^* = F(x_k) - F^* \leq C(1-\rho)^k(F(x_0) - F^*) = C(1-\rho)^{\frac{s}{T_M}}(F(x_0) - F^*) \leq C(1 - \frac{\rho}{T_M})^s(F(x_0) - F^*)$$

The first inequality is due to Th. 3.1 and the second is $(1-\rho)^x \leq 1 - x\rho, x \in [0,1]$. We concluded that the algorithm had a linear convergence rate which equals to

$$\tau_{\mathscr{A},F} = \rho/T_{\mathscr{M}} = \tilde{O}(\tau_M \frac{\sqrt{\mu}}{\sqrt{\mu + k}})$$

3.2 Analysis of Non-Strongly Convex Case

That is when $\mu = 0$, we took $\alpha_0 = \frac{\sqrt{5}-1}{2}$ and $\epsilon_k = \frac{2(F(x_0) - F^*)}{9(k+2)^{4+\eta}}, \eta > 0$ (which should be small enough) , the counterpart for Th.3.1 is Th.3.3, where by Algorithm1 we generates a sequence $(x_k)_{k\geq 0}$ such that, regardless of subproblem (5), we can get formula (11) as below. For subproblem (5), similarly, we got Proposition 3.4, where the upper bound for the number of iterations to update $x_{k-1} \to x_k$ is $T_{\mathscr{M}} log(k+2)$ for given algorithm $\mathscr{M}$. Thus, we got formula (12).

$$F(x_k) - F^* \leq \frac{8}{(k+2)^2}\left(\left(1 + \frac{2}{\eta}\right)^2 (F(x_0) - F^*) + \frac{\kappa}{2}\|x_0 - x^*\|^2\right). \tag{11}$$

$$F_s - F^* \leq \frac{8T_{\mathcal{M}}^2 \log^2(s)}{s^2}\left(\left(1 + \frac{2}{\eta}\right)^2 (F(x_0) - F^*) + \frac{\kappa}{2}\|x_0 - x^*\|^2\right). \tag{12}$$

4 Applications and Experiments

The Section 4 analyze the effect of catalyst algorithm combined with different other algorithms, especially SAG, SAGA, which has never been accelerated before that. The following graph explains the improvement of catalyst-acceleration algorithm.

	Comp. $\mu > 0$	Comp. $\mu = 0$	Catalyst $\mu > 0$	Catalyst $\mu = 0$
FG	$O\left(n\left(\frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(n\frac{L}{\varepsilon}\right)$	$\tilde{O}\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	$\tilde{O}\left(n\frac{L}{\sqrt{\varepsilon}}\right)$
SAG [24]	$O\left(\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$		$\tilde{O}\left(\sqrt{\frac{nL}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	
SAGA [6]				
Finito/MISO-Prox		not avail.		
SDCA [25]				
SVRG [27]	$O\left(\frac{L'}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$		$\tilde{O}\left(\sqrt{\frac{nL'}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	
Acc-FG [19]	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(n\frac{L}{\sqrt{\varepsilon}}\right)$	no acceleration	
Acc-SDCA [26]	$\tilde{O}\left(\sqrt{\frac{nL}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	not avail.		

Table 1: Comparison of rates of convergence, before and after the catalyst acceleration, resp. in the strongly-convex and non strongly-convex cases. **To simplify, we only present the case where** $n \leq L/\mu$ **when** $\mu > 0$. For all incremental algorithms, there is indeed no acceleration otherwise. The quantity L' for SVRG is the average Lipschitz constant of the functions f_i (see [27]).

We could see from the table that catalyst algorithm doesn't always accelerate the original algorithm, as an example, for SAG, improvements happen only when $L > n\mu$, or $L/\mu > n$.

Remember that is when the condition number of the objective function($k(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$) is very large, the ill-conditioned case. If we take $L = 1$, and $n = 10000$, then for the inequality to hold, we need a square penalty term parameter $\mu < 1e-4$ to see the difference of a linear square loss objective function.

In the authors' experiments, they also observed that for some data set and if $L/\mu > n$ wasn't satisfied, catalyst algorithms couldn't outperform the classical ones. But for those ill-conditioned problem, catalyst provided a much faster and stable solution.

5 A Simple Simulation on Homework Data Set

We also applied the catalyst acceleration algorithm to Stochastic Average Gradient (SAG) in the strongly convex case and present the simulation result in the Jupyter Notebook. The simulation is by Python and we used the data source accessible by http://www.di.ens.fr/~fbach/orsay2017/data_orsay_2017.mat , which is a binary classification mission. Note that the Lipschitz value L for this data set is not large enough so that we can easily distinguish the outperformance of catalyst-acceleration algorithm over classical SAG, yet we can still see some differences between the two algorithms.

Function F used in our simulation is $F(b) = \frac{1}{m}\sum_{i=1}^{m}(< x_i, b > - y_i)^2 + \frac{\mu}{2}\|b\|^2$, note that we take F as $\mu - strongly$ convex function. We calculated the Fenchel-Duality and the duality gap of $G(b) = F(b) + \frac{k}{2}\|b - z\|$ as an upper bound for estimating the distance to wanted optimal objective function value, so that we knew we can stop the

inner loop of catalyst algorithm when $G(b) - G(b^*) < duality\ gap < \epsilon_k$. Functions for calculating the duality gap and Graph 3 of duality gap — # passes can be found in the Appendix of this report, we saw that the duality gap almost reached 0 at last.

The restarting strategy is also used in our simulation, otherwise each time the error will jump high for starting each inner loop. In simulation, we have decided to take

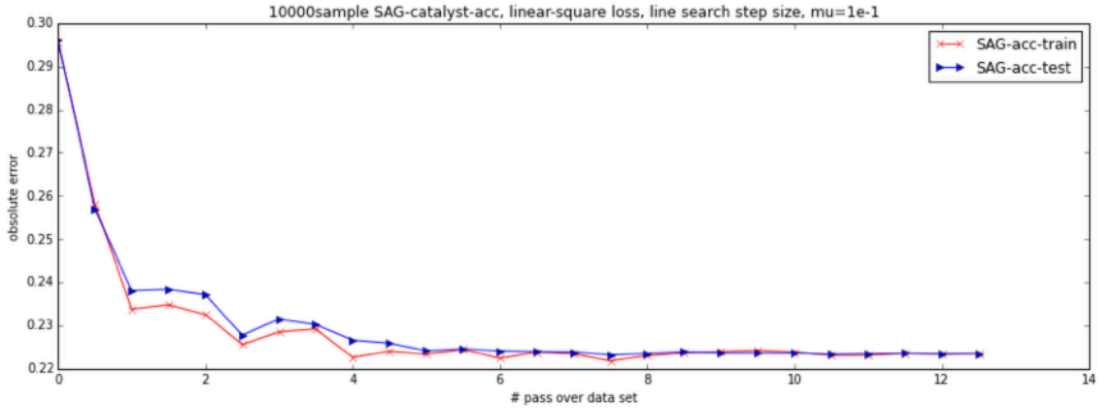
$$k = 2e-4,\ \mu = 1e-5,\ q = \frac{\mu}{\mu + k},\ \rho = 0.9\sqrt{q},\ \alpha_0 = 0.5(q - 1 + \sqrt{(1-q)^2 + 4}),$$

by which the simulation for catalyst worked much better. The loss function converged within 3 rounds and for each round there were about 2 to 6 passes of the data set, based on the parameters of the simulation.

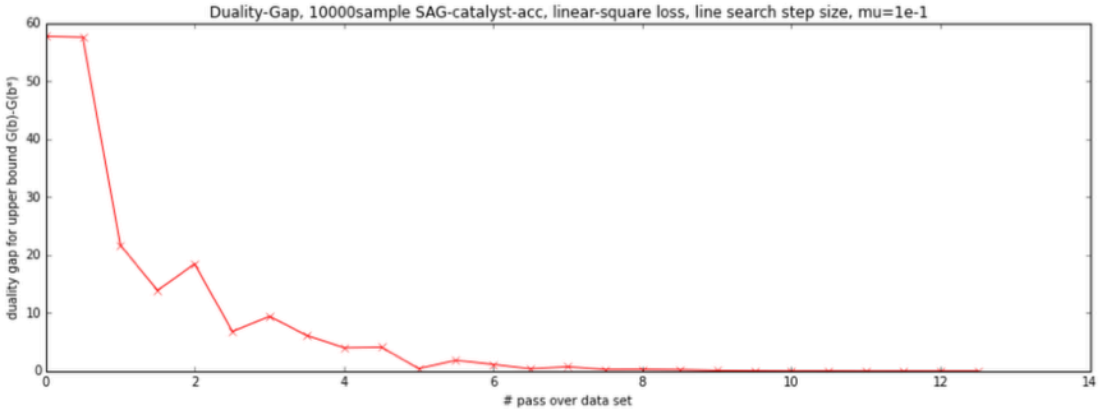
We could see (Graph 3) that, compared with classical SAG, on this data set catalyst still accelerated the SAG at the beginning and the blue line seems to have stabilized faster than classical SAG. However, as our data set is not ill-conditioned (our training data's L=1, very small), the benefits of catalyst are not very clear.

APPENDIX OF REPORT

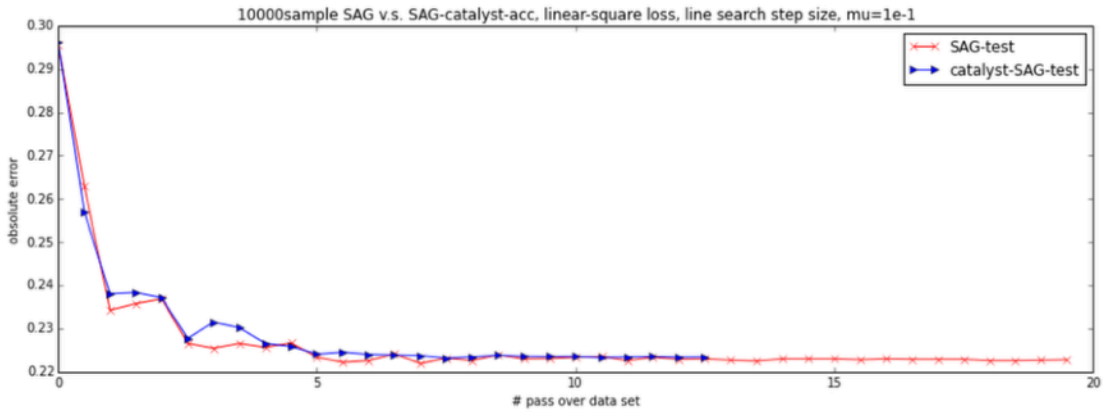
1 Graphs of Simulation



Graph1. SAG-Catalyst, train and test absolute error—# of passes of data set



Graph2. SAG-Catalyst, duality gap — # of passes of data set



Graph3. SAG-Catalyst, SAG-test error v.s. catalyst-SAG-test error

2 The Fenchel Duality-Gap

Take $G(b) = \frac{1}{m}\sum_{i=1}^{m}(< x_i, b > - y_i)^2 + \frac{\mu}{2}\|b\|^2 + \frac{k}{2}\|b - z\|$, and we denote $X = (x_1^T, x_2^T, \ldots, x_m^T)$, $y = (y_1, y_2, \ldots, y_m)$ as data features and data labels, each sample $x_i \in R^p$ and there are m samples in total.

Function for calculating Fenchel's duality of G(b) is

$$G^*(\phi) = -\frac{1}{\mu + k}\{2k < z, -X\phi > + \|X\phi\|^2 - k\mu\|z\|^2\} - \{\frac{m}{2}\|\phi\|^2 + < \phi, y > \}$$

And the Fenchel duality-gap is, $G(b) - G^*(\phi(b)) =$

$$\frac{1}{m}\|X^Tb - y\| + \frac{1}{2}\mu\|b\|^2 + \frac{k}{2}\|b - z\|^2 + 1/(mu + k)\{-\frac{k}{m}z.dot(Xerr) + \frac{1}{2m^2}\|X(X^Tb - y)\|^2 - \frac{k\mu}{2}\|z\|^2\} + \frac{1}{m} < X^Tb - y, y >$$

where $\phi(b) = \frac{1}{m}(X^Tb - y)$.

Codes can be found in the Jupyter notebook.