A Universal Catalyst for Gradient-Based Optimization

Julien Mairal
Inria, Grenoble

CIMI workshop, Toulouse, 2015
Collaborators

Hongzhou Lin

Zaid Harchaoui

Publication

Focus of this work

Minimizing large finite sums

Given some data, learn some model parameters $x$ in $\mathbb{R}^p$ by minimizing

$$
\min_{x \in \mathbb{R}^p} \left\{ F(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\},
$$

where each $f_i$ is smooth and convex and $\psi$ is a convex but not necessarily differentiable regularization penalty.

Goal of this work

- Design accelerated methods for minimizing large finite sums.
- Accelerate previously un-accelerated algorithms.
Why do large finite sums matter?

Empirical risk minimization

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\},$$

- Typically, $x$ represents **model parameters**.
- Each function $f_i$ measures the **fidelity** of $x$ to a data point.
- $\psi$ is a **regularization function** to prevent overfitting.

For instance, given training data $(y_i, z_i)_{i=1,...,n}$ with features $z_i$ in $\mathbb{R}^p$ and labels $y_i$ in $\{-1, +1\}$, $f_i$ may measure how far the prediction $\text{sign}(\langle z_i, x \rangle)$ is from the true label $y_i$.

This would be a **classification problem with a linear model**.
Why large finite sums matter?

A few examples

**Ridge regression:**
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - \langle x, z_i \rangle)^2 + \frac{\lambda}{2} \|x\|_2^2.
\]

**Linear SVM:**
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle x, z_i \rangle) + \frac{\lambda}{2} \|x\|_2^2.
\]

**Logistic regression:**
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y_i \langle x, z_i \rangle}\right) + \frac{\lambda}{2} \|x\|_2^2.
\]
Why does the composite problem matter?

A few examples

Ridge regression:
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - \langle x, z_i \rangle)^2 + \frac{\lambda}{2} \|x\|_2^2.
\]

Linear SVM:
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle x, z_i \rangle) + \frac{\lambda}{2} \|x\|_2^2.
\]

Logistic regression:
\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y_i \langle x, z_i \rangle}\right) + \frac{\lambda}{2} \|x\|_2^2.
\]

Regularization was called by Vladimir Vapnik “one of the first signs of the existence of intelligent inference”.

The **squared $l_2$-norm** penalizes large entries in $x$. 
Why does the composite problem matter?

A few examples

**Ridge regression:**

\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - \langle x, z_i \rangle)^2 + \lambda \|x\|_1.
\]

**Linear SVM:**

\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle x, z_i \rangle)^2 + \lambda \|x\|_1.
\]

**Logistic regression:**

\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y_i \langle x, z_i \rangle}\right) + \lambda \|x\|_1.
\]

When one knows in advance that \( x \) should be sparse, one should use a **sparsity-inducing** regularization such as the \( \ell_1 \)-norm.

[Chen et al., 1999, Tibshirani, 1996].
What do we mean by “acceleration”? 

Let us consider the composite problem 

$$ \min_{x \in \mathbb{R}^p} f(x) + \psi(x), $$

where $f$ is convex, differentiable with $L$-Lipschitz continuous gradient and $\psi$ is convex, but not necessarily differentiable.

The classical forward-backward/ISTA algorithm 

$$ x_k \leftarrow \arg\min_{x \in \mathbb{R}^p} \frac{1}{2} \left\| x - \left( x_{k-1} - \frac{1}{L} \nabla f(x_{k-1}) \right) \right\|^2_2 + \frac{1}{L} \psi(x). $$

- $f(x_k) - f^* = O(1/k)$ for convex problems;
- $f(x_k) - f^* = O((1 - \mu/L)^k)$ for $\mu$-strongly convex problems;

What do we mean by “acceleration”? 

Nesterov introduced in the 80’s an acceleration scheme for the gradient descent algorithm. It was generalized later to the composite setting.

**FISTA [Beck and Teboulle, 2009]**

\[
x_k \leftarrow \text{arg min}_{x \in \mathbb{R}^p} \frac{1}{2} \left\| x - \left( y_{k-1} - \frac{1}{L} \nabla f(y_{k-1}) \right) \right\|_2^2 + \frac{1}{L} \psi(x);
\]

Find \( \alpha_k > 0 \) s.t. \( \alpha_k^2 = (1 - \alpha_k) \alpha_{k-1}^2 + \frac{\mu}{L} \alpha_t; \)

\[
y_k \leftarrow x_{k-1} + \left( \frac{\alpha_{k-1}(1 - \alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k} \right) (x_t - x_{k-1}).
\]

- \( f(x_k) - f^* = O(1/k^2) \) for **convex** problems;
- \( f(x_k) - f^* = O((1 - \sqrt{\mu/L})^k) \) for \( \mu \)-**strongly convex** problems;

see also [Nesterov, 1983, 2004, 2013]
What do we mean by “acceleration”?

Remarks
- Acceleration works in many practical cases.
- FISTA is parameter-free thanks to simple line-search schemes.
- The original FISTA paper was for $\mu = 0$ and $\alpha_0 = (\sqrt{5} - 1)/2$.

Complexity analysis for large finite sums
To minimize a sum of $n$ functions with the guarantee $f(x_k) - f^* \leq \varepsilon$, the number of gradient evaluations is upper-bounded by

<table>
<thead>
<tr>
<th></th>
<th>$\mu &gt; 0$</th>
<th>$\mu = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISTA</td>
<td>$O \left( n \frac{L}{\mu} \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
<td>$O \left( \frac{nL}{\varepsilon} \right)$</td>
</tr>
<tr>
<td>FISTA</td>
<td>$O \left( n \sqrt{\frac{L}{\mu}} \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
<td>$O \left( \frac{nL}{\sqrt{\varepsilon}} \right)$</td>
</tr>
</tbody>
</table>
Can we do better for large finite sums?

For $n = 1$, no!
The rates are optimal for a “first-order local black box” [Nesterov, 2004].

For $n \geq 1$, yes! We need to design algorithms

- whose per-iteration computational complexity is smaller than $n$;
- whose convergence rate may be worse than FISTA....
- ...but with a better expected computational complexity.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISTA</td>
<td>$O\left(n\sqrt{\frac{L}{\mu}} \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
</tr>
<tr>
<td>SVRG, SAG, SAGA, SDCA, MISO, Finito</td>
<td>$O\left(\left(n + \frac{L}{\mu}\right) \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
</tr>
</tbody>
</table>

[Schmidt et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014a,b, Shalev-Shwartz and Zhang, 2012, Mairal, 2015, Zhang and Xiao, 2015]
Can we do even better for large finite sums?

SVRG, SAG, SAGA, SDCA, MISO, Finito improve upon FISTA when \( n \geq \sqrt{\frac{L}{\mu}} \), but they are not “accelerated” in the sense of Nesterov.

Without vs with acceleration

<table>
<thead>
<tr>
<th></th>
<th>( \mu &gt; 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVRG, SAG, SAGA, SDCA, MISO, Finito</td>
<td>( O \left( \left( n + \frac{L}{\mu} \right) \log \left( \frac{1}{\varepsilon} \right) \right) )</td>
</tr>
<tr>
<td>Acc-SDCA</td>
<td>( \tilde{O} \left( \left( n + \sqrt{n} \frac{L}{\mu} \right) \log \left( \frac{1}{\varepsilon} \right) \right) )</td>
</tr>
</tbody>
</table>

Remarks

- Acceleration occurs when \( \frac{L}{\mu} \geq n \).
- see [Agarwal and Bottou, 2015] for discussions about optimality.

Acc-SDCA is due to Shalev-Shwartz and Zhang [2014].
Outline of this presentation

Part I: Catalyst

Part II: Proximal MISO
Outline of this presentation

Part I: Catalyst

Part II: Proximal MISO
Main idea

Catalyst, a meta-algorithm

Consider an algorithm $\mathcal{M}$ that can minimize $F$ up to any accuracy.

- At iteration $k$, replace $F$ by a stabilized counterpart $G_k$ centered at the prox-center $y_k$.

$$G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_k\|^2,$$

and minimize $G_k$ up to accuracy $\varepsilon_k$, i.e., such that

$$G_k(x_k) - G_k^* \leq \varepsilon_k.$$ 

- Compute the next prox-center $y_k$ using an extrapolation step

$$y_k = x_k + \beta_k (x_k - x_{k-1}).$$

The choices of $\beta_k, \varepsilon_k, \kappa$ are driven by the theoretical analysis.

Catalyst is a wrapper of $\mathcal{M}$ that yields an accelerated algorithm $\mathcal{A}$. 
This work

Contributions

- **Generic acceleration scheme**, which applies to previously unaccelerated algorithms such as SVRG, SAG, SAGA, SDCA, MISO, or Finito, and which is not tailored to finite sums.
- Provides explicit **support to non-strongly convex objectives**.
- Complexity analysis for $\mu$-strongly convex objectives.
- Complexity analysis for non-strongly convex objectives.

Example of application

Garber and Hazan [2015] have used Catalyst to accelerate new principal component analysis algorithms based on convex optimization.
Sources of inspiration

In addition to accelerated proximal algorithms [Beck and Teboulle, 2009, Nesterov, 2013], there are two works that have inspired Catalyst.

The inexact accelerated proximal point algorithm of Güler [1992].
- Catalyst is a variant of inexact accelerated PPA.
- Complexity analysis for outer-loop only.

Accelerated SDCA of Shalev-Shwartz and Zhang [2014].
- Accelerated SDCA is an instance of inexact accelerated PPA.
- Complexity analysis limited to $\mu$-strongly convex objectives.

The complexity analysis is not just a theoretical exercise since it provides the values of $\kappa, \varepsilon_k, \beta_k$, which are required in concrete implementations.

Here, theoretical values match practical ones.
Other related works

Analysis of inexact accelerated algorithms.
- Accelerated PPA with various “inexactness” criterions [Salzo and Villa, 2012, He and Yuan, 2012], and with complexity analysis limited to outer-loop.
- Inexact accelerated forward-backward [Schmidt et al., 2011].

Unregularizing [Frostig et al., 2015].
- Acceleration based on inexact accelerated PPA.
- Complexity analysis limited to $\mu$-strongly convex objectives.

Randomized primal-dual gradient [Lan, 2015].
- Algorithm for minimizing large finite sums with “built-in” acceleration (no outer-loop).
- Primal-dual and game-theoretic interpretation.
Theoretical setting

Linear convergence of $\mathcal{M}$

Assume that $F$ is $\mu$-strongly convex. An algorithm $\mathcal{M}$ has a **linear convergence rate** if there exists $\tau_{\mathcal{M},F}$ in $(0, 1)$ and a constant $C_{\mathcal{M},F}$ in $\mathbb{R}$ such that

$$F(x_k) - F^* \leq C_{\mathcal{M},F}(1 - \tau_{\mathcal{M},F})^k. \quad (1)$$

For a given algorithm, $\tau_{\mathcal{M},F}$ depends usually on the **condition number** $L/\mu$, e.g., $\tau_{\mathcal{M},F} = \mu/L$ for ISTA.

Catalyst applies to algorithms that satisfy (1), which is classical for gradient-based approaches, yielding a method $\mathcal{A}$ with $\tau_{\mathcal{A},F} \geq \tau_{\mathcal{M},F}$. 

Julien Mairal, Inria

Catalyst
Catalyst action

Consider an algorithm $\mathcal{M}$ that can minimize $F$ up to any accuracy. At iteration $k$, replace $F$ by a stabilized counterpart $G_k$ centered at the prox-center $y_k$.

$$G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_k\|_2^2,$$

and minimize $G_k$ up to accuracy $\varepsilon_k$.

- If $\kappa \gg 1$, then minimizing $G_k$ is easy (the new condition number is $\frac{L+\kappa}{\mu+\kappa} \ll \frac{L}{\mu}$).
- If $\kappa \approx 0$, then $G_k$ is a good approximation of $F$. 

How does “acceleration” work?

Unfortunately, the literature does not provide any simple geometric explanation...
How does “acceleration” work?

Unfortunately, the literature does not provide any simple geometric explanation... but they are a few obvious facts, and a mechanism introduced by Nesterov, called “estimate sequence”.

Obvious fact

- Simple gradient descent steps are “blind” to the past iterates, and are based on a purely local model of the objective.
- Accelerated methods usually involve an extrapolation step
  \[ y_k = x_k + \beta_k (x_k - x_{k-1}) \]
  with \( \beta_k \) in \((0, 1)\).
How does “acceleration” work?

Unfortunately, the literature does not provide any simple geometric explanation... but they are a few obvious facts, and a mechanism introduced by Nesterov, called “estimate sequence”.

**Obvious fact**

- Simple gradient descent steps are “blind” to the past iterates, and are based on a **purely local** model of the objective.
- Accelerated methods usually involve an **extrapolation step**
  \[ y_k = x_k + \beta_k (x_k - x_{k-1}) \] with \( \beta_k \) in \( (0, 1) \).

Unfortunately, extrapolation only gives a vague intuition and does not explain why it leads to a concrete converging accelerated algorithms.
How does “acceleration” work?

If $f$ is $\mu$-strongly convex and $\nabla f$ is $L$-Lipschitz continuous

- $f(x) \leq f(x_{k-1}) + \nabla f(x_{k-1})^T(x - x_{k-1}) + \frac{L}{2} \|x - x_{k-1}\|^2_2$;
- $f(x) \geq f(x_{k-1}) + \nabla f(x_{k-1})^T(x - x_{k-1}) + \frac{\mu}{2} \|x - x_{k-1}\|^2_2$;
How does “acceleration” work?

If $\nabla f$ is $L$-Lipschitz continuous

$$f(x) \leq f(x_{k-1}) + \nabla f(x_{k-1})^T (x - x_{k-1}) + \frac{L}{2} \|x - x_{k-1}\|^2_2;$$

$$x_k = x_{k-1} - \frac{1}{L} \nabla f(x_{k-1}) \text{ (gradient descent step).}$$
How does “acceleration” work?

Gradient descent with step-size $1/L$ can be interpreted as iteratively minimizing local upper-bounds of the objective.

This is also the case for ISTA for minimizing $F(x) \triangleq f(x) + \psi(x)$:

$$x_k \leftarrow \arg \min_{x \in \mathbb{R}^p} f(x_{k-1}) + \nabla f(x_{k-1})^\top (x - x_{k-1}) + \frac{L}{2} \|x - x_{k-1}\|_2^2 + \psi(x).$$

Instead, accelerated gradient methods construct models of the objective with less conservative and less local principles (use of lower-bounds and upper-bounds, use of previous models to construct the current one).

These models, called estimate sequences, were introduced by Nesterov. They provide methodology for designing algorithms and proving their convergence.
How does “acceleration” work?

Definition of estimate sequence [Nesterov].

A pair of sequences \( (\varphi_k)_{t \geq 0} \) and \( (\lambda_k)_{t \geq 0} \), with \( \lambda_k \geq 0 \) and \( \varphi_k : \mathbb{R}^p \rightarrow \mathbb{R} \), is called an estimate sequence of function \( F \) if \( \lambda_k \rightarrow 0 \) and

for any \( x \in \mathbb{R}^p \) and all \( t \geq 0 \),

\[
\varphi_k(x) - F(x) \leq \lambda_k (\varphi_0(x) - F(x)).
\]

In addition, if for some sequence \( (x_k)_{k \geq 0} \) we have

\[
F(x_k) \leq \varphi_k^* \triangleq \min_{x \in \mathbb{R}^p} \varphi_k(x),
\]

then

\[
F(x_k) - F^* \leq \lambda_k (\varphi_0(x^*) - F^*),
\]

where \( x^* \) is a minimizer of \( F \).
How does “acceleration” work?

In summary, we need two properties

1. $\varphi_k(x) \leq (1 - \lambda_k)F(x) + \lambda_k \varphi_0(x)$;
2. $F(x_k) \leq \varphi^*_k \overset{\Delta}{=} \min_{x \in \mathbb{R}^p} \varphi_k(x)$.

Remarks

- $\varphi_k$ is neither an upper-bound, nor a lower-bound;
- Finding the right estimate sequence is often nontrivial.
How does “acceleration” work?

In summary, we need two properties

1. \( \varphi_k(x) \leq (1 - \lambda_k)F(x) + \lambda_k \varphi_0(x) \);

2. \( F(x_k) \leq \varphi^*_k \triangleq \min_{x \in \mathbb{R}^p} \varphi_k(x) \).

How to build an estimate sequence?

Define \( \varphi_k \) recursively

\[
\varphi_k(x) \triangleq (1 - \alpha_k)\varphi_{k-1}(x) + \alpha_k d_k(x),
\]

where \( d_k \) is a lower-bound, e.g., if \( F \) is smooth,

\[
d_k(x) \triangleq F(y_k) + \nabla F(y_k)^\top (x - y_k) + \frac{\mu}{2} \|x - y_k\|_2^2,
\]

Then, work hard to choose \( \alpha_k \) as large as possible, and \( y_k \) and \( x_k \) such that property 2 holds. Subsequently, \( \lambda_k = \prod_{t=1}^k (1 - \alpha_t) \).
Back to Catalyst

Catalyst consists of solving, up to accuracy $\varepsilon_k$, the sub-problems

$$G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_k\|^2_2,$$

Main challenges

- Find the right estimate sequence—that is, the right lower-bound.
- Introduce the right definition of approximate estimate sequence.
- Control the accumulation of errors $(\varepsilon_k)_{t \geq 0}$.
- Control the complexity of solving the sub-problems with $\mathcal{M}$.

Differences with [Güler, 1992]

- different estimate sequence and more rigorous convergence analysis of the outer-loop algorithm.
- weaker inexactness criterion which allows us to control the complexity of the inner-loop algorithm.
Catalyst, the algorithm

**Algorithm 1 Catalyst**

**input** initial estimate $x_0 \in \mathbb{R}^p$, parameters $\kappa$ and $\alpha_0$, sequence $(\varepsilon_k)_{t \geq 0}$, optimization method $\mathcal{M}$; initialize $q = \mu / (\mu + \kappa)$ and $y_0 = x_0$;

1: while the desired stopping criterion is not satisfied do

2: Find an approx. solution $x_k$ using $\mathcal{M}$ s.t. $G_k(x_k) - G_k^* \leq \varepsilon_k$

$$x_k \approx \arg\min_{x \in \mathbb{R}^p} \left\{ G_t(x) \triangleq F(x) + \frac{\kappa}{2} ||x - y_{k-1}||^2 \right\}$$

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

4: Compute

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

5: end while

**output** $x_k$ (final estimate).
Analysis for $\mu$-strongly convex objective functions

Convergence of outer-loop algorithm

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, the algorithm generates iterates $(x_k)_{t \geq 0}$ such that

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

Remarks

- Choice of $\rho$ can safely be set to $\rho = 0.9\sqrt{q}$ in practice.
- Choice of $(\varepsilon_k)_{k \geq 0}$ typically follows from a duality gap at $x_0$. When $F$ is non-negative, set $\varepsilon_k = (2/9)F(x_0)(1 - \rho)^k$. 

Julien Mairal, Inria

Catalyst 28/58
Analysis for $\mu$-strongly convex objective functions

Control of inner-loop complexity

Under the same assumptions, consider a method $\mathcal{M}$ generating iterates $(z_t)_{t \geq 0}$ for minimizing $G_k$ with linear convergence rate

$$G_k(z_t) - G^*_k \leq A(1 - \tau_M)^t(G_k(z_0) - G^*_k).$$

When $z_0 = x_{k-1}$, the precision $\varepsilon_k$ is reached with a number of iterations $T_M = \tilde{O}(1/\tau_M)$, where the notation $\tilde{O}$ hides some universal constants and some logarithmic dependencies in $\mu$ and $\kappa$. 
Analysis for \( \mu \)-strongly convex objective functions

Global computational complexity

By calling \( F_s \) the objective function value obtained after performing \( s = kT_M \) iterations of the method \( M \), the true convergence rate of the accelerated algorithm \( A \) is

\[
F_s - F^* = F \left( x, \frac{s}{T_M} \right) - F^* \leq C \left( 1 - \frac{\rho}{T_M} \right)^s (F(x_0) - F^*).
\] (2)

Algorithm \( A \) has a global linear rate of convergence with parameter

\[
\tau_{A,F} = \frac{\rho}{T_M} = \tilde{O}(\tau_M \sqrt{\mu} / \sqrt{\mu + \kappa}),
\]

where \( \tau_M \) typically depends on \( \kappa \) (the greater, the faster is \( M \)).

\( \kappa \) will be chosen to maximize the ratio \( \tau_M / \sqrt{\mu + \kappa} \).
Analysis for $\mu$-strongly convex objective functions

Global computational complexity

By calling $F_s$ the objective function value obtained after performing $s = kT_M$ iterations of the method $M$, the true convergence rate of the accelerated algorithm $A$ is

$$F_s - F^* = F\left(x_{\frac{s}{T_M}}\right) - F^* \leq C \left(1 - \frac{\rho}{T_M}\right)^s (F(x_0) - F^*). \quad (2)$$

Algorithm $A$ has a global linear rate of convergence with parameter

$$\tau_{A,F} = \rho / T_M = \tilde{O}(\tau_M \sqrt{\mu} / \sqrt{\mu + \kappa}),$$

where $\tau_M$ typically depends on $\kappa$ (the greater, the faster is $M$).

\[
\text{e.g., } \kappa = L - 2\mu \text{ when } \tau_M = \frac{\mu + \kappa}{L + \kappa} \Rightarrow \tau_A = \tilde{O} \left(\sqrt{\frac{\mu}{L}}\right).
\]
Analysis for non-strongly convex objective functions

Convergence of outer-loop algorithm , $\mu = 0$

Choose $\alpha_0 = \frac{\sqrt{5} - 1}{2}$ and

$$\epsilon_k = \frac{2(F(x_0) - F^*)}{9(k + 2)^{4+\eta}} \quad \text{with } \eta > 0.$$

Then, the meta-algorithm generates iterates $(x_k)_{k \geq 0}$ such that

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

(3)

Remarks

- the meta-algorithm achieves the optimal rate of convergence of first-order methods;
- this result disproves a conjecture of Salzo and Villa [2012].
Analysis for non-strongly convex objective functions

Control of inner-loop complexity

Assume that $F$ has bounded level sets. Under the same assumptions, let us consider a method $\mathcal{M}$ generating iterates $(z_t)_{t\geq 0}$ for minimizing the function $G_k$ with linear convergence rate:

$$ G_k(z_t) - G_k^* \leq A (1 - \tau_{\mathcal{M}})^t (G_k(z_0) - G_k^*). $$

Then, there exists $T_{\mathcal{M}} = \tilde{O}(1/\tau_{\mathcal{M}})$, such that for any $k \geq 1$, solving $G_k$ with initial point $x_{k-1}$ requires at most $T_{\mathcal{M}} \log(k + 2)$ iterations of $\mathcal{M}$. 
Analysis for non-strongly convex objective functions

Global computational complexity

To produce $x_k$, $\mathcal{M}$ is called at most $kT\mathcal{M} \log(k + 2)$ times. Using the global iteration counter $s = kT\mathcal{M} \log(k + 2)$, we get

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

If $\mathcal{M}$ is a first-order method, this rate is near-optimal, up to a logarithmic factor, when compared to the optimal rate $O(1/s^2)$, which may be the price to pay for using a generic acceleration scheme.

$\kappa$ will be chosen to maximize the ratio $\tau\mathcal{M}/\sqrt{L + \kappa}$. 
Catalyst in practice

General strategy and application to randomized algorithms

Calculating the iteration-complexity decomposes into three steps:

1. When $F$ is $\mu$-strongly convex, find $\kappa$ that maximizes the ratio $\tau_{\mathcal{M},G_k}/\sqrt{\mu + \kappa}$ for algorithm $\mathcal{M}$. When $F$ is non-strongly convex, maximize instead the ratio $\tau_{\mathcal{M},G_k}/\sqrt{L + \kappa}$.

2. Compute the upper-bound of the number of outer iterations $k_{\text{out}}$ using the theorems.

3. Compute the upper-bound of the expected number of inner iterations

   $$\max_{k=1,...,k_{\text{out}}} \mathbb{E}[T_{\mathcal{M},G_k}(\varepsilon_k)] \leq k_{\text{in}},$$

Then, the expected iteration-complexity denoted Comp. is given by

$$\text{Comp} \leq k_{\text{in}} \times k_{\text{out}}.$$
Applications

Deterministic and Randomized Incremental Gradient methods

- Stochastic Average Gradient (SAG and SAGA) [Schmidt et al., 2013, Defazio et al., 2014a];
- Finito and MISO [Mairal, 2015, Defazio et al., 2014b];
- Semi-Stochastic/Mixed Gradient [Konečnỳ et al., 2014, Zhang et al., 2013];
- Stochastic Dual coordinate Ascent [Shalev-Shwartz and Zhang, 2012];
- Stochastic Variance Reduced Gradient [Xiao and Zhang, 2014].

But also, randomized coordinate descent methods, and more generally first-order methods with linear convergence rates.
Applications

Expected computational complexity in the regime $n \leq L/\mu$ when $\mu > 0$,

<table>
<thead>
<tr>
<th></th>
<th>$\mu &gt; 0$</th>
<th>$\mu = 0$</th>
<th>Catalyst $\mu &gt; 0$</th>
<th>Cat. $\mu = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG</td>
<td>$O\left(n \left(\frac{L}{\mu}\right) \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>$O\left(n \frac{L}{\varepsilon}\right)$</td>
<td>$\tilde{O}\left(n \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>$\tilde{O}\left(n \frac{L}{\sqrt{\varepsilon}}\right)$</td>
</tr>
<tr>
<td>SAG</td>
<td>$O\left(\frac{L}{\mu} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>$O\left(n \frac{L}{\varepsilon}\right)$</td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>$\tilde{O}\left(\frac{nL}{\sqrt{\varepsilon}}\right)$</td>
</tr>
<tr>
<td>SAGA</td>
<td></td>
<td></td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td></td>
</tr>
<tr>
<td>Finito/MISO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDCA</td>
<td></td>
<td>$\tilde{O}\left(\frac{L'}{\mu} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVRG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc-FG</td>
<td>$O\left(n \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>$O\left(n \frac{L}{\sqrt{\varepsilon}}\right)$</td>
<td></td>
<td>no acceleration</td>
</tr>
<tr>
<td>Acc-SDCA</td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experiments with MISO/SAG/SAGA

Restarting

The theory tells us to restart $\mathcal{M}$ with $x_{k-1}$. For SDCA/Finito/MISO, the theory tells us to use instead $x_{k-1} + \frac{\kappa}{\mu + \kappa}(y_{k-1} - y_{k-2})$. We also tried this as a heuristic for SAG and SAGA.

One-pass heuristic

constrain $\mathcal{M}$ to always perform at most $n$ iterations; we call this variant AMISO2 for MISO, whereas AMISO1 refers to the regular “vanilla” accelerated variant; idem to accelerate SAG and SAGA.
Experiments with MISO/SAG/SAGA

\(\ell_2\)-logistic regression formulation

Given some data \((y_i, z_i)\), with \(y_i \in \{-1, +1\}\) and \(z_i \in \mathbb{R}^p\), minimize

\[
\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i x^\top z_i}) + \frac{\mu}{2} \|x\|_2^2,
\]

\(\mu\) is the regularization parameter and the strong convexity modulus.

Datasets

<table>
<thead>
<tr>
<th>name</th>
<th>rcv1</th>
<th>real-sim</th>
<th>covtype</th>
<th>ocr</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>781 265</td>
<td>72 309</td>
<td>581 012</td>
<td>2 500 000</td>
<td>250 000</td>
</tr>
<tr>
<td>(p)</td>
<td>47 152</td>
<td>20 958</td>
<td>54</td>
<td>1 155</td>
<td>500</td>
</tr>
</tbody>
</table>
Experiments without strong convexity, $\mu = 0$

![Graphs showing objective function value for different number of passes on data.](image)

**Figure**: Objective function value for different number of passes on data.

**Conclusions**

- SAG, SAGA are accelerated when they do not perform well already;
- $\text{AMISO2} \geq \text{AMISO1} \text{ (vanilla)}$, MISO does not apply.
Experiments without strong convexity, $\mu = 10^{-1}/n$

Figure: Relative duality gap (log-scale) for different number of passes on data.

Conclusions

- SAG, SAGA are not always accelerated, but often.
- AMISO2, AMISO1 $\gg$ MISO.
Experiments without strong convexity, $\mu = 10^{-3}/n$

---

**Figure**: Relative duality gap (log-scale) for different number of passes on data.

**Conclusions**

- Same conclusions as $\mu = 10^{-1}/n$;
- $\mu$ is so small that (unaccelerated) MISO becomes numerically unstable.
General conclusions about Catalyst

Summary: lots of nice features

- Simple acceleration scheme with broad application range.
- Recover near-optimal rates for known algorithms.
- Effortless implementation.

... but also lots of unsolved problems

- Acceleration occurs when $n \leq L/\mu$; otherwise, the “unaccelerated” complexity $O(n \log(1/\varepsilon))$ seems unbeatable.
- $\mu$ is an estimate of the true strong convexity parameter $\mu' \geq \mu$.
- $\mu$ is the global strong convexity parameter, not a local one $\mu^* \geq \mu$.
- When $n \leq L/\mu$, but $n \geq L/(\mu' \text{ or } \mu^*)$, a method $\mathcal{M}$ that adapts to the unknown strong convexity may be impossible to accelerate.
- The optimal restart for $\mathcal{M}$ is not yet fully understood.
Ideas of the proofs

Main theorem

Let us denote

\[ \lambda_k = \prod_{i=0}^{k-1} (1 - \alpha_i), \] (4)

where the \( \alpha_i \)'s are defined in Catalyst. Then, the sequence \((x_k)_{k \geq 0}\) satisfies

\[ F(x_k) - F^* \leq \lambda_k \left( \sqrt{S_k} + 2 \sum_{i=1}^{k} \sqrt{\frac{\epsilon_i}{\lambda_i}} \right)^2, \] (5)

where \( F^* \) is the minimum value of \( F \) and

\[ S_k = F(x_0) - F^* + \frac{\gamma_0}{2} \|x_0 - x^*\|^2 + \sum_{i=1}^{k} \frac{\epsilon_i}{\lambda_i} \] where \[ \gamma_0 = \frac{\alpha_0 \left( (\kappa + \mu)\alpha_0 - \mu \right)}{1 - \alpha_0}, \] (6)

where \( x^* \) is a minimizer of \( F \).
Ideas of the proofs

Then, the theorem will be used with the following lemma to control the convergence rate of the sequence \((\lambda_k)_{k \geq 0}\), whose definition follows the classical use of estimate sequences. This will provide us convergence rates both for the strongly convex and non-strongly convex cases.

Lemma 2.2.4 from Nesterov [2004]

If in the quantity \(\gamma_0\) defined in (6) satisfies \(\gamma_0 \geq \mu\), then the sequence \((\lambda_k)_{k \geq 0}\) from (4) satisfies

\[
\lambda_k \leq \min \left\{ (1 - \sqrt{q})^k, \frac{4}{\left(2 + k \sqrt{\frac{\gamma_0}{\kappa + \mu}}\right)^2} \right\},
\]

(7)

where \(q \triangleq \mu / (\mu + \kappa)\).
Ideas of proofs

Step 1: build an approximate estimate sequence

- Remember that in general, we build $\varphi_k$ from $\varphi_{k-1}$ as follows

$$\varphi_k(x) \triangleq (1 - \alpha_k)\varphi_{k-1}(x) + \alpha_k d_k(x),$$

where $d_k$ is a lower bound.

- Here, a natural lower bound would be

$$F(x) \geq d_k(x) \triangleq F(x^*_k) + \langle \kappa (y_{k-1} - x^*_k), x - x^*_k \rangle + \frac{\mu}{2} \|x - x^*_k\|^2,$$

where $x^*_k \triangleq \arg\min_{x \in \mathbb{R}^p} \left\{ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_{k-1}\|^2 \right\}$.

- But $x^*_k$ is unknown! Then, use instead $d'_k(x)$ defined as

$$d'_k(x) \triangleq F(x_k) + \langle \kappa (y_{k-1} - x_k), x - x_k \rangle + \frac{\mu}{2} \|x - x_k\|^2.$$
Ideas of proofs

Step 2: Relax the condition $F(x_k) \leq \phi_k^\star$.

- We can show that Catalyst generates iterates $(x_k)_{k \geq 0}$ such that

$$F(x_k) \leq \phi_k^\star + \xi_k,$$

where the sequence $(\xi_k)_{k \geq 0}$ is defined by $\xi_0 = 0$ and

$$\xi_k = (1 - \alpha_{k-1})(\xi_{k-1} + \varepsilon_k - (\kappa + \mu) \langle x_k - x_k^\star, x_{k-1} - x_k \rangle).$$

- The sequences $(\alpha_k)_{k \geq 0}$ and $(\gamma_k)_{k \geq 0}$ are chosen in such a way that all the terms involving $y_{k-1} - x_k$ are cancelled.

- We will control later the quantity $x_k - x_k^\star$ by strong convexity of $G_k$:

$$\frac{\kappa + \mu}{2} \|x_k - x_k^\star\|_2^2 \leq G_k(x_k) - G_k^\star \leq \varepsilon_k.$$
Ideas of proofs

Step 3: Control how this errors sum up together.
  - Do cumbersome calculus.
Outline

Part I: Catalyst

Part II: Proximal MISO
Original motivation

Given some data, learn some model parameters $x$ in $\mathbb{R}^p$ by minimizing

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\},$$

where each $f_i$ may be nonsmooth and nonconvex.

The original MISO algorithm is an incremental extension of the majorization-minimization principle [Lange et al., 2000].

Publications

Majorization-minimization principle

- Iteratively minimize locally tight upper bounds of the objective.
- The objective monotonically decreases.
- Under some assumptions, we get similar convergence rates as gradient-based approaches for convex problems.
Algorithm 2 Incremental scheme MISO

**input** $x_0 \in \mathbb{R}^p$; $T$ (number of iterations).
1: Choose surrogates $g^0_i$ of $f_i$ near $x_0$ for all $i$;
2: for $k = 1, \ldots, K$ do
3: Randomly pick up one index $\hat{i}_k$ and choose a surrogate $g^{k}_{\hat{i}_k}$ of $f_{\hat{i}_k}$
   near $x_{k-1}$. Set $g^k_i \triangleq g^{k-1}_i$ for $i \neq \hat{i}_k$;
4: Update the solution:
   
   $$x_k \in \arg\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} g^k_i(x).$$
5: end for

**output** $x_K$ (final estimate);
Incremental Optimization: MISO

Update rule with basic upper bounds

We want to minimize \( \frac{1}{n} \sum_{i=1}^{n} f_i(x) \), where the \( f_i \)'s are smooth.

\[
x_k \leftarrow \arg \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(y_i^k) + \nabla f_i(y_i^k)^\top (x - y_i^k) + \frac{L}{2} \| x - y_i^k \|_2^2
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} y_i^k - \frac{1}{Ln} \sum_{i=1}^{n} \nabla f_i(y_i^k).
\]

At iteration \( k \), randomly draw one index \( \hat{i}_k \), and update \( y_{\hat{i}_k}^k \leftarrow x_k \).

Remarks

- replace \( (1/n) \sum_{i=1}^{n} y_i^k \) by \( x_{k-1} \) yields SAG [Schmidt et al., 2013].
- replace \( (1/L) \) by \( (1/\mu) \) for strongly convex problems is close to a variant of SDCA [Shalev-Shwartz and Zhang, 2012].
Incremental Optimization: MISO$_\mu$.

Update rule with lower bounds???

We want to minimize $\frac{1}{n} \sum_{i=1}^{n} f_i(x)$, where the $f_i$'s are smooth.

$$x_k = \arg \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(y_i^k) + \nabla f_i(y_i^k)^\top (x - y_i^k) + \frac{\mu}{2} \|x - y_i^k\|_2^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} y_i^k - \frac{1}{\mu n} \sum_{i=1}^{n} \nabla f_i(y_i^k).$$

Remarks

- Requires strong convexity.
- Use a counter-intuitive minorization-minimization principle.
- Close to a variant of SDCA [Shalev-Shwartz and Zhang, 2012].
- Much faster than the basic MISO (faster rate).
Incremental Optimization: \( \text{MISO}_\mu \).

In the first part of this presentation, what we have called MISO is the algorithm that uses \( \frac{1}{(\mu n)} \) step-sizes (sorry for the confusion).

To minimize \( F(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) \), MISO\( \mu \) has the following guarantees

**Proposition [Mairal, 2015]**

When the functions \( f_i \) are \( \mu \)-strongly convex, differentiable with \( L \)-Lipschitz gradient, and non-negative, MISO\( \mu \) satisfies

\[
\mathbb{E}[F(x_k) - F^*] \leq \left(1 - \frac{1}{3n}\right)^k nf^*,
\]

under the condition \( n \geq 2L/\mu \).

**Remarks**

- When \( n \leq 2L/\mu \), the algorithm may diverge;
- When \( \mu \) is very small, numerical stability is an issue.
- The condition \( f_i \geq 0 \) does not really matter.
Proximal MISO [Lin, Mairal, and Harchaoui, 2015]

Main goals

- Remove the condition $n \leq 2L/\mu$;
- Allow a composite term $\psi$:

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\},$$

Starting points

MISO$\mu$ is iteratively updating/minimizing a lower-bound of $F$

$$x_k \leftarrow \arg\min_{x \in \mathbb{R}^p} \left\{ D_k(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} d_i^k(x) \right\},$$

[Lin, Mairal, and Harchaoui, 2015].
Proximal MISO

Adding the proximal term

\[ x_t \leftarrow \arg\min_{x \in \mathbb{R}^p} \left\{ D_k(x) \triangleq \frac{1}{n} \sum_{i=1}^n d_i^k(x) + \psi(x) \right\}, \]

Remove the condition \( n \geq 2L/\mu \)

For \( i = \hat{i}_k \),

\[ d_i^k(x) = (1-\delta)d_i^{k-1}(x) + \delta \left( f_i(x_{k-1}) + \langle \nabla f_i(x_{k-1}), x - x_{k-1} \rangle + \frac{\mu}{2} \|x - x_{k-1}\|^2 \right) \]

Remarks

- the original MISO\( \mu \) uses \( \delta = 1 \). To get rid of the condition \( n \geq 2L/\mu \), proximal MISO uses instead \( \delta = \min \left( 1, \frac{\mu n}{2(L-\mu)} \right) \).

- variant “5” of SDCA [Shalev-Shwartz and Zhang, 2012] is identical with another value \( \delta = \frac{\mu n}{L+\mu n} \) in \((0, 1)\).
Proximal MISO

Convergence of MISO-Prox

Let \((x_k)_{k \geq 0}\) be obtained by MISO-Prox, then

\[
\mathbb{E}[F(x_k)] - F^* \leq \frac{1}{\tau} (1 - \tau)^{k+1} (F(x_0) - D_0(x_0)) \quad \text{with} \quad \tau \geq \min \left\{ \frac{\mu}{4L}, \frac{1}{2n} \right\}.
\]

Furthermore, we also have fast convergence of the certificate

\[
\mathbb{E}[F(x_k) - D_k(x_k)] \leq \frac{1}{\tau} (1 - \tau)^k (F^* - D_0(x_0)).
\]

Differences with SDCA

- The construction is **primal**. The proof of convergence and the algorithm do not use duality, while SDCA is a dual ascent technique.
- \(D_k(x_k)\) is a lower-bound of \(F^*\); it plays the same role as the dual in SDCA, but is **easier to evaluate**.
Conclusions

- Relatively simple algorithm, with simple convergence proof, and simple optimality certificate.
- Catalyst not only accelerates it, but also stabilizes it numerically, with the parameter $\delta = 1$.
- Close to SDCA, but without duality.


References II


References III


References IV


References V


