A Universal Catalyst for Gradient-Based Optimization

Julien Mairal
Inria, Grenoble

Gatsby seminar
Collaborators

Hongzhou Lin

Zaid Harchaoui

Publication

Focus of this work

Minimizing large finite sums

Consider the minimization of a large sum of convex functions

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

where each \( f_i \) is **smooth and convex** and \( \psi \) is a convex but not necessarily differentiable penalty.

Goal of this work

- Design accelerated methods for minimizing large finite sums.
- Give a generic acceleration scheme which can apply to 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 $\{-1, +1\}$, we may want to predict $y_i$ by $\text{sign}(\langle z_i, x \rangle)$. Functions $f_i$ measures how far the prediction is from the true label.

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 $\ell_2$-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].

Julien Mairal  Catalyst
How to minimize a large sum composite problem?

Two major challenges

- **Non-differentiable regularization penalty.**
  Exclude existing solver such as MOSEK, CPLEX, etc.

- **Large-scale dimension.**
  Exclude higher-order (Newton) methods.

This leads us to first-order gradient-based methods.
Gradient descent methods

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;

Accelerated gradient descent methods

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]

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

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

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

- \( f(x_k) - f^* = O(1/k^2) \) for convex problems;
- \( f(x_k) - f^* = O((1 - \sqrt{\mu/L})^k) \) for \( \mu \)-strongly convex problems;
- Acceleration works in many practical cases.

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

Complexity analysis for large finite sums

Since \( f \) is a sum of \( n \) functions, computing \( \nabla f \) requires computing \( n \) gradients \( \nabla f_i \). The complexity to reach an \( \varepsilon \)-solution is given below.

<table>
<thead>
<tr>
<th></th>
<th>( \mu &gt; 0 )</th>
<th>( \mu = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISTA</td>
<td>( O \left( \frac{nL}{\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>

Remarks

- \( \varepsilon \)-solution means here \( f(x_k) - f^* \leq \varepsilon \).
- For \( n = 1 \), the rates of FISTA are optimal for a “first-order local black box” [Nesterov, 2004].
- For \( n > 1 \), the sum structure of \( f \) is not exploited.
Can we do better for large finite sums?

Several **randomized** algorithms are designed with one $\nabla f_i$ computed per iteration, which yields a better **expected computational complexity**.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\mu &gt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISTA</td>
<td>$O\left(n\sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
</tr>
<tr>
<td>SVRG, SAG, SAGA, SDCA, MISO, Finito</td>
<td>$O\left(\max\left(n, \frac{L}{\mu}\right) \log \left(\frac{1}{\epsilon}\right)\right)$</td>
</tr>
</tbody>
</table>

SVRG, SAG, SAGA, SDCA, MISO, Finito improve upon FISTA when

$$\max\left(n, \frac{L}{\mu}\right) \leq n\sqrt{\frac{L}{\mu}} \Leftrightarrow \sqrt{\frac{L}{\mu}} \leq n,$$

but they are not “accelerated” in the sense of Nesterov.

[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?

Without vs with acceleration

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Without Acceleration</th>
<th>With Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISTA</td>
<td>$O \left( n \sqrt{\frac{L}{\mu}} \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
<td>$\tilde{O} \left( \max (n, \sqrt{n \frac{L}{\mu}}) \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
</tr>
<tr>
<td>SVRG, SAG, SAGA, SDCA, MISO, Finito</td>
<td>$O \left( \max (n, \frac{L}{\mu}) \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
<td>$\tilde{O} \left( \max (n, \sqrt{n \frac{L}{\mu}}) \log \left( \frac{1}{\varepsilon} \right) \right)$</td>
</tr>
</tbody>
</table>

Acc-SDCA is due to Shalev-Shwartz and Zhang [2014].

- Acceleration occurs when $n \leq \frac{L}{\mu}$.
- See [Agarwal and Bottou, 2015] for discussions about optimality.

Challenge: can we accelerate these algorithms by a universal scheme for both convex and strongly convex objectives?
Catalyst is coming
Main idea

Catalyst, a meta-algorithm

Given an algorithm $\mathcal{M}$ that can solve a convex problem ”appropriately”.

- At iteration $k$, rather than minimizing $F$, we use $\mathcal{M}$ to minimize a function $G_k$, defined as follows,

\[
G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_{k-1}\|^2_2,
\]

up to accuracy $\varepsilon_k$, i.e., such that $G_k(x_k) - G_k^* \leq \varepsilon_k$.

- Then 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.
Main idea

Catalyst, a meta-algorithm

Given an algorithm $\mathcal{M}$ that can solve a convex problem "appropriately".

- At iteration $k$, rather than minimizing $F$, we use $\mathcal{M}$ to minimize a function $G_k$, defined as follows,

\[
G_k(x) \triangleq F(x) + \frac{\kappa}{2} \|x - y_{k-1}\|^2,
\]

up to accuracy $\varepsilon_k$, i.e., such that $G_k(x_k) - G_k^* \leq \varepsilon_k$.

- Then 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 $\mathcal{A}$. 
Sources of inspiration

In addition to accelerated proximal algorithms [Beck and Teboulle, 2009, Nesterov, 2013], several works 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 with non practical inexactness criterium.

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.
Sources of inspiration

In addition to accelerated proximal algorithms [Beck and Teboulle, 2009, Nesterov, 2013], several works 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 with non practical inexactness criterium.

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.

Other related work
[Chambolle and Pock, 15].
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.
Appropriate $\mathcal{M} = \text{Linear convergence rate when } \mu > 0$

Linear convergence rate

Consider a strongly convex minimization problem

$$\min_{z \in \mathbb{R}^p} H(z).$$

We say that an algorithm $\mathcal{M}$ has a linear convergence rate if $\mathcal{M}$ generates a sequence of iterates $(z_t)_{t \in \mathbb{N}}$ such that there exists $\tau_{\mathcal{M},H}$ in $(0, 1)$ and a constant $C_{\mathcal{M},H}$ in $\mathbb{R}$ satisfying

$$H(z_t) - H^* \leq C_{\mathcal{M},H}(1 - \tau_{\mathcal{M},H})^t. \quad (1)$$

- $\tau_{\mathcal{M},H}$ depends usually on the condition number $L/\mu$, e.g., $\tau_{\mathcal{M},H} = \mu/L$ for ISTA and $\tau_{\mathcal{M},H} = \sqrt{\mu/L}$ for FISTA.
- $C_{\mathcal{M},H}$ depends usually on $H(z_0) - H^*$. 
Appropriate $\mathcal{M} = \text{Linear convergence rate when } \mu > 0$

**Linear convergence rate**

Consider a *strongly convex* minimization problem

$$
\min_{z \in \mathbb{R}^p} H(z).
$$

We say that an algorithm $\mathcal{M}$ has a **linear convergence rate** if $\mathcal{M}$ generates a sequence of iterates $(z_t)_{t \in \mathbb{N}}$ such that there exists $\tau_{\mathcal{M},H}$ in $(0, 1)$ and a constant $C_{\mathcal{M},H}$ in $\mathbb{R}$ satisfying

$$
H(z_t) - H^* \leq C_{\mathcal{M},H}(1 - \tau_{\mathcal{M},H})^t.
$$

(1)

**Important message:** we do not make any assumption for non strongly convex objectives. It is possible that $\mathcal{M}$ is not even defined for $\mu = 0.$
Catalyst action

\[ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \| x - y_{k-1} \|^2 \],

- \( G_k \) is always strongly convex as long as \( F \) is convex.
- When \( F \) is strongly convex, the condition number of \( G_k \) is better than that of \( F \) since \( \frac{L + \kappa}{\mu + \kappa} < \frac{L}{\mu} \).
Catalyst action

\[ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \| x - y_{k-1} \|^2, \]

- \( G_k \) is always strongly convex as long as \( F \) is convex.
- When \( F \) is strongly convex, the condition number of \( G_k \) is better than that of \( F \) since \( \frac{L + \kappa}{\mu + \kappa} < \frac{L}{\mu} \).

**Minimizing \( G_k \) is easier than minimizing \( F \) !**
Catalyst action

\[ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \| x - y_{k-1} \|_2^2, \]

- \( G_k \) is always strongly convex as long as \( F \) is convex.
- When \( F \) is strongly convex, the condition number of \( G_k \) is better than that of \( F \) since \( \frac{L + \kappa}{\mu + \kappa} < \frac{L}{\mu} \).

**Minimizing \( G_k \) is easier than minimizing \( F \) !**

- If \( \kappa \gg 1 \), then minimizing \( G_k \) is easy;
- If \( \kappa \approx 0 \), then \( G_k \) is a good approximation of \( F \).

We will choose \( \kappa \) to optimize the computational complexity.
Convergence analysis

An analysis in two stages

\[ G_k(x) \triangleq F(x) + \frac{\kappa}{2} \| x - y_{k-1} \|_2^2, \]

\( x_k \) is a approximate minimizer of \( G_k \) such that \( G_k(x_k) - G_k^* \leq \epsilon_k \).

- Outer loop: once we obtain the sequence \( (x_k)_{k \in \mathbb{N}} \), what can we say about the convergence rate of \( F(x_k) - F^* \)?
  \( \Rightarrow \) Wisely choose \( (y_k) \) and control the accumulation of errors.

- Inner loop: how much effort do we need to obtain a \( x_k \) with accuracy \( \epsilon_k \)?
  \( \Rightarrow \) Wisely choose the starting point.
Choice of \((y_k)_{k \in \mathbb{N}}\)

**Extrapolation**

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

- This update is identical to Nesterov’s accelerated gradient descent or FISTA.
- Unfortunately, the literature does not provide any simple geometric explanation why it yields an acceleration...
- The construction is purely theoretical by using a mechanism introduced by Nesterov, called “estimate sequence”.

Julien Mairal

Catalyst
How does “acceleration” work?

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

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

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

\[
    f(x) \leq f(x_{k-1}) + \nabla f(x_{k-1})^\top (x - x_{k-1}) + \frac{L}{2} \|x - x_{k-1}\|^2;
\]

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

Definition of estimate sequence [Nesterov].

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

- \(\lambda_k \rightarrow 0\);
- \(\varphi_k(x) \leq (1 - \lambda_k)F(x) + \lambda_k \varphi_0(x)\), for any \(k, x\);
- There exists a sequence \((x_k)_{k \geq 0}\) such that

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

Remarks

- \(\varphi_k\) is neither an upper-bound, nor a lower-bound;
- Finding the right estimate sequence is often nontrivial.
Convergence of outer-loop algorithm

Analysis for $\mu$-strongly convex objective functions

Choose $\alpha_0 = \sqrt{q}$ with $q = \mu/(\mu + \kappa)$ and

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

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

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

In practice

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

Analysis for non-strongly convex objective functions, $\mu = 0$

Choose $\alpha_0 = (\sqrt{5} - 1)/2$ 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

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

(2)

In practice

- Choice of $\eta$ can be set to $\eta = 0.1$. 
How many iterates of $\mathcal{M}$ do we need to obtain $x_k$?

Control of inner-loop complexity

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

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

Then by choosing $z_0 = x_{k-1}$, the precision $\varepsilon_k$ is reached with at most

- A constant number of iterations $T_\mathcal{M}$ when $\mu > 0$;
- A logarithmic increasing number of iterations $T_\mathcal{M} \log(k + 2)$ when $\mu = 0$.

where $T_\mathcal{M} = \tilde{O}(1/\tau_\mathcal{M})$ is independent of $k$. 
Global computational complexity

Analysis for $\mu$-strongly convex objective functions

The global convergence rate of the accelerated algorithm $\mathcal{A}$ is

$$F_s - F^* \leq C \left( 1 - \frac{\rho}{T_M} \right)^s (F(x_0) - F^*). \tag{3}$$

where $F_s$ is the objective function value obtained after performing $s = kT_M$ iterations of the method $\mathcal{M}$. As a result,

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

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

$\kappa$ will be chosen to maximize the ratio $\tau_M/\sqrt{\mu + \kappa}$. 

Julien Mairal
Catalyst 28/45
Global computational complexity

Analysis for $\mu$-strongly convex objective functions

The global convergence rate of the accelerated algorithm $A$ is

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

(3)

where $F_s$ is the objective function value obtained after performing $s = kT_M$ iterations of the method $M$. As a result,

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

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

e.g., $\kappa = L - 2\mu$ when $\tau_M = \frac{\mu + \kappa}{L + \kappa} \Rightarrow \tau_A = \tilde{O}\left(\sqrt{\frac{\mu}{L}}\right)$.  

Julien Mairal  Catalyst  28/45
Global computational complexity

Analysis for non-strongly convex objective functions

The global convergence rate of the accelerated algorithm $\mathcal{A}$ is

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

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.

$k$ will be chosen to maximize the ratio $\tau_{\mathcal{M}}/\sqrt{L + k}$. 
Applications

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

<table>
<thead>
<tr>
<th>Method</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(n \frac{L}{\sqrt{\varepsilon}}\right))</td>
</tr>
<tr>
<td>SAGA</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(n \frac{L}{\sqrt{\varepsilon}}\right))</td>
</tr>
<tr>
<td>Finito/MISO</td>
<td>( O\left(\frac{L}{\mu} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>NA</td>
<td>( \tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>( \tilde{O}\left(n \frac{L}{\sqrt{\varepsilon}}\right))</td>
</tr>
<tr>
<td>SDCA</td>
<td>( O\left(\frac{L'}{\mu} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>NA</td>
<td>( \tilde{O}\left(\sqrt{\frac{nL'}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>( \tilde{O}\left(n \frac{L}{\sqrt{\varepsilon}}\right))</td>
</tr>
<tr>
<td>SVRG</td>
<td>( O\left(\frac{L'}{\mu} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>NA</td>
<td>( \tilde{O}\left(\sqrt{\frac{nL'}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>( \tilde{O}\left(n \frac{L}{\sqrt{\varepsilon}}\right))</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>no acceleration</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>( \tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\varepsilon}\right)\right))</td>
<td>( \tilde{O}\left(n \frac{L}{\sqrt{\varepsilon}}\right))</td>
</tr>
</tbody>
</table>
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 with MISO/SAG/SAGA

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.**

**Restarting**

The theory tells us to restart $M$ 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 $M$ to always perform at most $n$ iterations in inner loop; we call this variant AMISO2 for MISO, whereas AMISO1 refers to the regular “vanilla” accelerated variant; idem to accelerate SAG and SAGA.
Experiments without strong convexity, \( \mu = 0 \)

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

**Conclusions**

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

![Graphs showing relative duality gap for different datasets and algorithms.](image)

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

**Conclusions**

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

![Graphs showing relative duality gap for different datasets with varying $\mu/L$](image)

**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 $n \geq L/(\mu'$ 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.
Thank you for your attention!
Algorithm 1 Catalyst

**Input** initial estimate $x_0 \in \mathbb{R}^p$, parameters $\kappa$ and $\alpha_0$, sequence $(\varepsilon_k)_{k \geq 0}$, 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. $G_k(x_k) - G^*_k \leq \varepsilon_k$
3: Compute $\alpha_k \in (0, 1)$ from equation $\alpha_k^2 = (1 - \alpha_k)\alpha_{k-1}^2 + q\alpha_k$;
4: Compute $y_k = x_k + \beta_k(x_k - x_{k-1})$ with $\beta_k = \frac{\alpha_{k-1}(1 - \alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k}$.
5: end while

**Output** $x_k$ (final estimate).
Ideas of the proofs

Main theorem

Let us denote

$$\lambda_k = \prod_{i=0}^{k-1} (1 - \alpha_i),$$  \hspace{1cm} (4)

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

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

where $F^*$ is the minimum value of $F$ and

$$S_k = F(x_0) - F^* + \frac{\gamma_0}{2} \|x_0 - x^*\|^2 + \sum_{i=1}^{k} \frac{\epsilon_i}{\lambda_i} \text{ where } \gamma_0 = \frac{\alpha_0 ((\kappa + \mu)\alpha_0 - \mu)}{1 - \alpha_0},$$  \hspace{1cm} (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

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

where \(q \triangleq \frac{\mu}{\mu + \kappa}\).
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

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

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 $F(x_k) \leq \varphi_k^\ast$.

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

\[ F(x_k) \leq \phi_k^* + \xi_k, \]

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

\[ \xi_k = (1 - \alpha_{k-1})(\xi_{k-1} + \varepsilon_k - (\kappa + \mu)(x_k - x_k^*, x_{k-1} - x_k)). \]

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

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

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

Step 3: Control how this errors sum up together.

- Do cumbersome calculus.
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

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

Then, the expected iteration-complexity denoted $\text{Comp.}$ 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.
Appendix on 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

\[ f(\theta) \]
\[ g(\theta) \]

- 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.
Incremental optimization: MISO

Algorithm 2 Incremental scheme MISO

**input** $x_0 \in \mathbb{R}^p$; $T$ (number of iterations).

1. Choose surrogates $g_i^0$ 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 $g_{\hat{i}_k}^k$ of $f_{\hat{i}_k}$ near $x_{k-1}$. Set $g_i^k \triangleq g_i^{k-1}$ for $i \neq \hat{i}_k$;
4. Update the solution:

$$x_k \in \arg \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} g_i^k(x).$$

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: MISO\(\mu\).

In the first part of this presentation, what we have called MISO is the algorithm that uses \(1/(\mu n)\) 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

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

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 \((0, 1)\).
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 \textit{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 \textit{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 I


References II


References V


