Generic Acceleration Schemes for Gradient-Based Optimization

Hongzhou Lin\textsuperscript{1}, Julien Mairal\textsuperscript{1}, Zaid Harchaoui\textsuperscript{2}

\textsuperscript{1}Inria, Grenoble \hspace{1cm} \textsuperscript{2}University of Washington

Séminaire d’optimisation
Paris, 2018
An alternate title: Acceleration by Smoothing
An alternate title: 
Acceleration by Smoothing

Another one: 
Recent Variants of the Inexact Proximal Point Algorithm
Collaborators

Hongzhou Lin  Zaid Harchaoui  Dima Drusvyatskiy  Courtney Paquette

Publications and pre-prints


Main motivation

Minimizing large finite sums

Consider the minimization of a large sum of convex functions

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

where each $f_i$ is \textit{L-smooth and convex} and \psi is a convex regularization penalty but not necessarily differentiable.

Motivation

Our goal is to accelerate existing algorithms

- with Nesterov’s principles;
- with Quasi-Newton heuristics;
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 \((y_i, z_i)_{i=1,\ldots,n}\) with features \( z_i \) in \( \mathbb{R}^p \) and labels \( y_i \) in \( \{-1, +1\} \), we may want to predict \( y_i \) by sign(\( \langle z_i, x \rangle \)). The functions \( f_i \) measure 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:**  \[ \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i \langle x, z_i \rangle} \right) + \frac{\lambda}{2} \|x\|_2^2. \]

The **squared \( \ell_2\)-norm** penalizes large entries in \( x \).

Julien Mairal
Generic Acceleration
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].
Part I: How to address finite-sum problems?
How to minimize a large finite sum of functions?

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

assuming here that the problem is \( \mu \)-strongly convex.

We consider several alternatives

- Batch first-order methods (ISTA, FISTA).
- Stochastic first-order methods (SGD, mirror descent).
- Incremental first-order methods (SAG, SAGA, SDCA, MISO, ...).
- Quasi-Newton approaches (L-BFGS).
(Batch) gradient descent methods

Let us consider the composite problem

$$\min_{x \in \mathbb{R}^p} \{ f(x) = f_0(x) + \psi(x) \} ,$$

where $f_0$ is convex, differentiable with $L$-Lipschitz continuous gradient and $\psi$ is convex, but not necessarily differentiable.

The classical forward-backward/ISTA algorithm

$$x_k \leftarrow \arg \min_{x \in \mathbb{R}^p} \frac{1}{2} \left\| x - \left( x_{k-1} - \frac{1}{L} \nabla f_0(x_{k-1}) \right) \right\|_2^2 + \frac{1}{L} \psi(x).$$

- $f(x_k) - f^* = O(1/k)$ 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

\[ x_k \leftarrow \arg \min_{x \in \mathbb{R}^p} \frac{1}{2} \left\| x - \left( y_{k-1} - \frac{1}{L} \nabla f_0(y_{k-1}) \right) \right\|^2_2 + \frac{1}{L} \psi(x); \]

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]
Stochastic gradient descent methods

... or the recent return of Robins and Monroe, 1951. Consider

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

At iteration $k$, select at random an index $i_k$, and perform the update

$$x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1})$$
... or the recent return of Robins and Monroe, 1951. Consider

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

At iteration $k$, select at random an index $i_k$, and perform the update

$$x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1}) \quad \text{(note that} \quad \mathbb{E}[\nabla f_{i_k}(x_{k-1})] = \nabla f(x_{k-1})\text{)}.$$
Stochastic gradient descent methods

... or the recent return of Robins and Monroe, 1951. Consider

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

At iteration $k$, select at random an index $i_k$, and perform the update

$$
x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1}) \quad \text{(note that } \mathbb{E}[\nabla f_{i_k}(x_{k-1})] = \nabla f(x_{k-1})\text{)}.
$$

Main features vs. batch

- **Complexity per-iteration is $n$ times smaller**;
Stochastic gradient descent methods

... or the recent return of Robins and Monroe, 1951. Consider

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

At iteration $k$, select at random an index $i_k$, and perform the update

$$x_k \leftarrow x_{k-1} - \eta_k \nabla f_i(x_{k-1}) \quad \text{(note that } \mathbb{E}[\nabla f_i(x_{k-1})] = \nabla f(x_{k-1})\text{)}.$$

Main features vs. batch

- **Complexity per-iteration is $n$ times smaller**;
- Convergence rate is slower: at most $O(1/k)$ for strongly-convex problems and $O(1/\sqrt{k})$ for convex ones, see [Nemirovski et al., 2009];
Stochastic gradient descent methods

... or the recent return of Robins and Monroe, 1951. Consider

\[
\min_{x \in \mathbb{R}^p} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}.
\]

At iteration \( k \), select at random an index \( i_k \), and perform the update

\[
x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1}) \quad \text{(note that } \mathbb{E}[\nabla f_{i_k}(x_{k-1})] = \nabla f(x_{k-1})\text{)}.
\]

Main features vs. batch

- **Complexity per-iteration is** \( n \) **times smaller**;
- Convergence rate is slower: at most \( O(1/k) \) for strongly-convex problems and \( O(1/\sqrt{k}) \) for convex ones, see [Nemirovski et al., 2009];
- variants are **compatible with prox** \( \psi \), e.g., [Duchi et al., 2011].
Stochastic gradient descent methods

... or the recent return of Robins and Monroe, 1951. Consider

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

At iteration $k$, select at random an index $i_k$, and perform the update

$$x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1}) \quad \text{(note that } \mathbb{E}[\nabla f_{i_k}(x_{k-1})] = \nabla f(x_{k-1})\text{)}).$$

Main features vs. batch

- **Complexity per-iteration is $n$ times smaller;**
- Convergence rate is slower: at most $O(1/k)$ for strongly-convex problems and $O(1/\sqrt{k})$ for convex ones, see [Nemirovski et al., 2009];
- variants are **compatible with prox** $\psi$, e.g., [Duchi et al., 2011].
- Sometimes a bit difficult to tune. When well tuned, the speed-up to obtain a solution with moderate accuracy may be huge.
Incremental gradient descent methods

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

Several randomized algorithms are designed with one $\nabla f_i$ computed per iteration, with fast convergence rates, e.g., SAG [Schmidt et al., 2017]:

$$x_k \leftarrow x_{k-1} - \frac{\gamma}{L_n} \sum_{i=1}^{n} y_i^k \text{ with } y_i^k = \begin{cases} \nabla f_i(x_{k-1}) & \text{if } i = i_k \\ y_i^{k-1} & \text{otherwise} \end{cases}.$$
Incremental gradient descent methods

\[
\min_{x \in \mathbb{R}^p} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}.
\]

Several randomized algorithms are designed with one \( \nabla f_i \) computed per iteration, with fast convergence rates, e.g., SAG [Schmidt et al., 2017]:

\[
x_k \leftarrow x_{k-1} - \gamma \sum_{i=1}^{n} y_i^k \quad \text{with} \quad y_i^k = \begin{cases} \nabla f_i(x_{k-1}) & \text{if } i = i_k \\
y_i^{k-1} & \text{otherwise} \end{cases}.
\]

See also SVRG, SAGA, SDCA, MISO, Finito...

Some of these algorithms perform updates of the form

\[
x_k \leftarrow x_{k-1} - \eta_k g_k \quad \text{with} \quad \mathbb{E}[g_k] = \nabla f(x_{k-1}),
\]

but \( g_k \) has lower variance than in SGD.

[Schmidt et al., 2017, Xiao and Zhang, 2014, Defazio et al., 2014a,b, Shalev-Shwartz and Zhang, 2012, Mairal, 2015, Zhang and Xiao, 2015]
Incremental gradient descent methods

These methods achieve low (worst-case) complexity in expectation. The number of gradients evaluations to ensure $f(x_k) - f^* \leq \varepsilon$ is

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISTA</td>
<td>$\mu &gt; 0$</td>
</tr>
<tr>
<td></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>$O\left(\max\left(n, \frac{L}{\mu}\right) \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
</tr>
</tbody>
</table>
Incremental gradient descent methods

These methods achieve low \textbf{(worst-case)} complexity in expectation. The number of gradients evaluations to ensure $f(x_k) - f^* \leq \varepsilon$ is

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

Main features vs. stochastic gradient descent

- Same complexity per-iteration (but higher memory footprint).
Incremental gradient descent methods

These methods achieve low (worst-case) complexity in expectation. The number of gradients evaluations to ensure $f(x_k) - f^* \leq \varepsilon$ is

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

Main features vs. stochastic gradient descent

- Same complexity per-iteration (but higher memory footprint).
- Faster convergence (exploit the finite-sum structure).
Incremental gradient descent methods

These methods achieve low (worst-case) complexity in expectation. The number of gradients evaluations to ensure \( f(x_k) - f^* \leq \varepsilon \) is

\[
\begin{array}{|c|c|}
\hline
\text{FISTA} & \mu > 0 \\
\hline
& O \left( n \sqrt{\frac{L}{\mu}} \log \left( \frac{1}{\varepsilon} \right) \right) \\
\hline
\text{SVRG, SAG, SAGA, SDCA, MISO, Finito} & O \left( \max \left( n, \frac{L}{\mu} \right) \log \left( \frac{1}{\varepsilon} \right) \right) \\
\hline
\end{array}
\]

Main features vs. stochastic gradient descent

- Same complexity per-iteration (but higher memory footprint).
- **Faster convergence** (exploit the finite-sum structure).
- **Less parameter tuning** than SGD.
Incremental gradient descent methods

These methods achieve low \textbf{(worst-case)} complexity in expectation. The number of gradients evaluations to ensure $f(x_k) - f^* \leq \varepsilon$ is

<table>
<thead>
<tr>
<th>Method</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>$O\left(\max\left(n, \frac{L}{\mu}\right) \log \left(\frac{1}{\varepsilon}\right)\right)$</td>
</tr>
</tbody>
</table>

Main features vs. stochastic gradient descent

- Same complexity per-iteration (but higher memory footprint).
- **Faster convergence** (exploit the finite-sum structure).
- **Less parameter tuning** than SGD.
- Some variants are compatible with composite term $\psi$. 
Part II: Catalyst
An old idea

Old idea: Smooth the function and then optimize.

The Moreau-Yosida envelope

Given $f : \mathbb{R}^d \to \mathbb{R}$ a convex function, the Moreau-Yosida envelope of $f$ is the function $F : \mathbb{R}^d \to \mathbb{R}$ defined as

$$F(x) = \min_{w \in \mathbb{R}^d} \left\{ f(w) + \frac{\kappa}{2} \| w - x \|^2 \right\}.$$ 

The proximal operator $p(x)$ is the unique minimizer of the problem.
The Moreau-Yosida regularization

\[ F(x) = \min_{w \in \mathbb{R}^d} \left\{ f(w) + \frac{\kappa}{2} \|w - x\|^2 \right\}. \]

Basic properties [see Lemaréchal and Sagastizábal, 1997]

- Minimizing \( f \) and \( F \) is equivalent in the sense that

\[ \min_{x \in \mathbb{R}^d} F(x) = \min_{x \in \mathbb{R}^d} f(x), \]

and the solution set of the two problems coincide with each other.

- \( F \) is continuously differentiable even when \( f \) is not and

\[ \nabla F(x) = \kappa (x - p(x)). \]

In addition, \( \nabla F \) is Lipschitz continuous with parameter \( L_F = \kappa \).

- If \( f \) is \( \mu \)-strongly convex then \( F \) is also strongly convex with parameter \( \mu_F = \frac{\mu \kappa}{\mu + \kappa} \).
The Moreau-Yosida regularization

\[ F(x) = \min_{w \in \mathbb{R}^d} \left\{ f(w) + \frac{\kappa}{2} \| w - x \|^2 \right\}. \]

Basic properties [see Lemaréchal and Sagastizábal, 1997]

- Minimizing \( f \) and \( F \) is equivalent in the sense that

  \[ \min_{x \in \mathbb{R}^d} F(x) = \min_{x \in \mathbb{R}^d} f(x), \]

  and the solution set of the two problems coincide with each other.

- \( F \) is continuously differentiable even when \( f \) is not and

  \[ \nabla F(x) = \kappa (x - p(x)). \]

  In addition, \( \nabla F \) is Lipschitz continuous with parameter \( L_F = \kappa \).

\( F \) enjoys nice properties: smoothness, (strong) convexity and we can control its condition number \( 1/q = 1 + \kappa/\mu \).
The proximal point algorithm

A naive approach consists of *minimizing the smoothed objective* $F$ *instead of* $f$ *with a method designed for smooth optimization.*

Consider indeed

$$x_{k+1} = x_k - \frac{1}{\kappa} \nabla F(x_k).$$

By rewriting the gradient $\nabla F(x_k)$ as $\kappa (x_k - p(x_k))$, we obtain

$$x_{k+1} = p(x_k) = \arg \min_{w \in \mathbb{R}^p} \left\{ f(w) + \frac{\kappa}{2} \| w - x_k \|^2 \right\}.$$ 

This is exactly the **proximal point algorithm** [Martinet, 1970, Rockafellar, 1976].
The accelerated proximal point algorithm

Consider now

\[ x_{k+1} = y_k - \frac{1}{\kappa} \nabla F(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k), \]

where \( \beta_{k+1} \) is a Nesterov-like extrapolation parameter. We may now rewrite the update using the value of \( \nabla F \), which gives:

\[ x_{k+1} = p(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k) \]

This is the accelerated proximal point algorithm of Güler [1992].
The accelerated proximal point algorithm

Consider now

\[ x_{k+1} = y_k - \frac{1}{\kappa} \nabla F(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k), \]

where \( \beta_{k+1} \) is a Nesterov-like extrapolation parameter. We may now rewrite the update using the value of \( \nabla F \), which gives:

\[ x_{k+1} = p(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k) \]

This is the accelerated proximal point algorithm of Güler [1992].

Remarks

- \( F \) may be better conditioned than \( f \) when \( 1 + \kappa/\mu \leq L/\mu \);
- Computing \( p(y_k) \) has a cost!
A fresh look at Catalyst [Lin, Mairal, and Harchaoui, 2015]

Catalyst is a particular accelerated proximal point algorithm with inexact gradients [Güler, 1992].

\[ x_{k+1} \approx p(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k) \]

The quantity \( x_{k+1} \) is obtained by using an optimization method \( M \) for approximately solving:

\[
\begin{align*}
    x_{k+1} \approx \arg \min_{w \in \mathbb{R}^p} \left\{ f(w) + \frac{\kappa}{2} \| w - y_k \|^2 \right\},
\end{align*}
\]

Catalyst provides Nesterov’s acceleration to \( M \) with...

- **restart strategies** for solving the sub-problems;
- **global complexity analysis** resulting in theoretical acceleration;
- **optimal balancing between outer and inner computations**.

see also [Frostig et al., 2015, Schmidt et al., 2011, Salzo and Villa, 2012, Devolder et al., 2014, Shalev-Shwartz and Zhang, 2016]
This work

Contributions

- **Generic acceleration scheme**, which applies to algorithms $\mathcal{M}$ that have **linear convergence rates** for strongly convex problems.
- Provides explicit **support to non-strongly convex objectives**.
- Complexity analysis for $\mu$-strongly convex objectives.
- Complexity analysis for non-strongly convex objectives.
- Extension to **non-convex optimization** by Paquette, Lin, Drusvyatskiy, Mairal, and Harchaoui [2017].
Requirements on $\mathcal{M}$

Linear convergence

- Say a sub-problem consists of minimizing $h$; we want $\mathcal{M}$ to produce a sequence of iterates $(z_t)_{t \geq 0}$ with **linear convergence rate**

$$h(z_t) - h^* \leq C_M (1 - \tau_M)^t (h(z_0) - h^*),$$

which may possibly hold only **in expectation** if $\mathcal{M}$ is randomized.

- **No assumption** is made on the behavior of $\mathcal{M}$ for **non-strongly convex problems**.

- Variants may be allowed when linear convergence is stated in terms of dual certificate.
When do we stop the method $\mathcal{M}$?

Three strategies to balance outer and inner computations

(a) use a **pre-defined sequence** $(\varepsilon_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the sub-problems $\min h_k$ satisfies

$$h_k(z_t) - h_k^* \leq \varepsilon_k.$$

(b) use a **pre-defined sequence** $(\delta_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the sub-problems $\min h_k$ satisfies

$$h_k(z_t) - h_k^* \leq \delta_k \|z_t - y_k\|^2.$$

(c) use a **pre-defined budget** $T_M$ of iterations of the method $\mathcal{M}$.
When do we stop the method $\mathcal{M}$?

Three strategies to balance outer and inner computations

(a) use a **pre-defined sequence** $(\varepsilon_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the sub-problems $\min h_k$ satisfies

$$h_k(z_t) - h_k^* \leq \varepsilon_k.$$ 

(b) use a **pre-defined sequence** $(\delta_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the sub-problems $\min h_k$ satisfies

$$h_k(z_t) - h_k^* \leq \frac{\delta_k}{2} \|z_t - y_k\|^2.$$ 

(c) use a **pre-defined budget** $T_{\mathcal{M}}$ of iterations of the method $\mathcal{M}$.

Remark

- (c) implies (a) and requires $T_{\mathcal{M}}$ to be larger than necessary in practice; it leads to the simplest and most effective strategies.
When do we stop the method $\mathcal{M}$?

Three strategies for $\mu$-strongly convex objectives $f$

(a) use

$$
\varepsilon_k = \frac{1}{2} C (1 - \rho)^{k+1} \quad \text{with} \quad C \geq f(x_0) - f^* \quad \text{and} \quad \rho < \sqrt{q}.
$$

where $q$ is the inverse of the condition number of $F$: $q = \frac{\mu}{(\mu + \kappa)}$

(b) use

$$
\delta_k = \frac{\sqrt{q}}{2 - \sqrt{q}}.
$$

(c) use a **pre-defined budget** $T_\mathcal{M}$ of iterations of the method $\mathcal{M}$ for solving each sub-problem with

$$
T_\mathcal{M} = \frac{1}{\tau_\mathcal{M}} \log \left( 19 C_\mathcal{M} \frac{L + \kappa}{\kappa} \right). \quad \text{(be more aggressive in practice)}
$$
When do we stop the method $\mathcal{M}$?

Three strategies for $\mu = 0$

(a) use

$$\varepsilon_k = \frac{f(x_0) - f^*}{2(k + 1)^{4+\gamma}} \text{ with } \gamma > 0.$$  

(b) use

$$\delta_k = \frac{1}{(k + 1)^2}.$$  

(c) use a **pre-defined budget** $T_k$ of iterations of the method $\mathcal{M}$ for solving each sub-problem $h_k$ with

$$T_k = O(\log(k)) \text{ (use a constant in practice)}$$
Other implementation details

See the arXiv paper for

- Nesterov’s extrapolation parameters (fairly standard).
- restart strategies for solving the sub-problems.
Other implementation details

See the arXiv paper for
- Nesterov’s extrapolation parameters (fairly standard).
- restart strategies for solving the sub-problems.

**Spoiler: optimal balance for inner/outer computations**

To choose $\kappa$, maximize
$$
\frac{\tau_M}{\sqrt{\mu + \kappa}}.
$$

Remember that $\tau_M$ drives the convergence rate for the sub-problems

$$
\frac{h(w_t) - h^*}{h(w_0) - h^*} \leq C_M (1 - \tau_M)^t (\frac{h(w_0) - h^*}{C_M}).
$$

For the standard gradient descent method, use $\kappa = L - 2\mu$. 
Outer-loop convergence analysis

With strong convexity

Using strategy (a),

\[ f(x_k) - f^* \leq C(1 - \rho)^{k+1}(f(x_0) - f^*) \quad \text{with} \quad \rho < \sqrt{q}, \]

and a similar result holds for (b).

Without strong convexity

Using strategy (b),

\[ f(x_k) - f^* \leq \frac{4\kappa \|x_0 - x^*\|^2}{(k + 1)^2}. \]

and a similar result holds for (a).
Inner-loop convergence analysis

Using appropriate restart strategies, the inner-loop stopping criterions are satisfied after $T_k$ iterations, where

$$T_k = \tilde{O} \left( \frac{1}{\tau_M} \right) \quad \text{when} \quad \mu > 0,$$

and

$$T_k = \tilde{O} \left( \frac{\log(k)}{\tau_M} \right) \quad \text{when} \quad \mu = 0.$$

The $\tilde{O}$ hides logarithmic quantities in $\mu, \kappa$ and universal constants.
Global complexity analysis

By combining the two previous strategies, we obtain that the guarantee $f(x_k) - f^* \leq \varepsilon$ is achieved after $N$ iterations of the method $\mathcal{M}$, where

$$N = \tilde{O} \left( \frac{1}{\tau \mathcal{M} \sqrt{q}} \log \left( \frac{1}{\varepsilon} \right) \right) \quad \text{when} \quad \mu > 0,$$

and

$$N = \tilde{O} \left( \frac{1}{\tau \mathcal{M}} \sqrt{\frac{\kappa}{\varepsilon}} \log \left( \frac{1}{\varepsilon} \right) \right) \quad \text{when} \quad \mu = 0.$$

Similar results hold also for randomized algorithms.
Global complexity analysis

By combining the two previous strategies, we obtain that the guarantee
\( f(x_k) - f^* \leq \varepsilon \) is achieved after \( N \) iterations of the method \( \mathcal{M} \), where

\[
N = \tilde{O} \left( \frac{1}{\tau \mathcal{M} \sqrt{q}} \log \left( \frac{1}{\varepsilon} \right) \right) \quad \text{when} \quad \mu > 0,
\]

and

\[
N = \tilde{O} \left( \frac{1}{\tau \mathcal{M}} \sqrt{\frac{\kappa}{\varepsilon}} \log \left( \frac{1}{\varepsilon} \right) \right) \quad \text{when} \quad \mu = 0.
\]

Similar results hold also for randomized algorithms.

Theoretical choice of \( \kappa \)

maximize

\[
\frac{\tau \mathcal{M}}{\sqrt{\mu + \kappa}}.
\]

For gradient descent, \( \tau \mathcal{M} = \frac{\mu + \kappa}{L + \kappa} \Rightarrow \kappa = L - 2\mu \Rightarrow \frac{1}{\tau \mathcal{M} \sqrt{q}} \leq 2 \sqrt{\frac{L}{\mu}} \)
**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>$O\left(n \left(\frac{L}{\mu}\right) \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td>$O\left(n \frac{L}{\epsilon}\right)$</td>
<td>$\tilde{O}\left(n \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td>$\tilde{O}\left(n \sqrt{\frac{L}{\epsilon}}\right)$</td>
</tr>
<tr>
<td>SAG</td>
<td></td>
<td>$O\left(n \frac{L}{\epsilon}\right)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAGA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finito/MISO</td>
<td>$O\left(\frac{L}{\mu} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td></td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\epsilon}}\right)$</td>
</tr>
<tr>
<td>SDCA</td>
<td></td>
<td>NA</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>$O\left(n \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td>$O\left(n \sqrt{\frac{L}{\epsilon}}\right)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc-SDCA</td>
<td>$\tilde{O}\left(\sqrt{\frac{nL}{\mu}} \log \left(\frac{1}{\epsilon}\right)\right)$</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no acceleration</td>
<td></td>
</tr>
</tbody>
</table>
Part III: QNing
Limited-Memory BFGS (L-BFGS)

Pros

- one of the largest practical success of smooth optimization.
Limited-Memory BFGS (L-BFGS)

Pros

- one of the largest practical success of smooth optimization.

Cons

- worst-case convergence rates for strongly-convex functions are linear, but much worse than the gradient descent method.
- proximal variants typically requires solving many times

\[
\min_{x \in \mathbb{R}^d} \frac{1}{2} (x - z) B_k (z - z) + \psi (x).
\]
- no guarantee of approximating the Hessian.
An old idea (again)

Old idea: Smooth the function and then optimize.

- The strategy appears in early work about variable metric bundle methods. [Chen and Fukushima, 1999, Fukushima and Qi, 1996, Mifflin, 1996, Fuentes, Malick, and Lemaréchal, 2012, Burke and Qian, 2000] ...

The Moreau-Yosida envelope

Given $f : \mathbb{R}^d \rightarrow \mathbb{R}$ a convex function, the Moreau-Yosida envelope of $f$ is the function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ defined as

$$F(x) = \min_{w \in \mathbb{R}^d} \left\{ f(w) + \frac{\kappa}{2} \| w - x \|^2 \right\} .$$

The proximal operator $p(x)$ is the unique minimizer of the problem.
QNing

Main recipe

- L-BFGS applied to the smoothed objective $F$ with inexact gradients [see Friedlander and Schmidt, 2012].
- Inexact gradients are obtained by solving sub-problems using a first-order optimization method $\mathcal{M}$;
- Ideally, $\mathcal{M}$ is able to adapt to the problem structure (finite sum, composite regularization).
- Replace L-BFGS steps by proximal point steps if no sufficient decrease is estimated $\Rightarrow$ no line search on $F$;
Obtaining inexact gradients

**Algorithm**  Procedure ApproxGradient

**input** Current point \( x \) in \( \mathbb{R}^d \); smoothing parameter \( \kappa > 0 \).

1. Compute the approximate mapping using an optimization method \( M \):
   \[
   z \approx \arg \min_{w \in \mathbb{R}^d} \left\{ h(w) \triangleq f(w) + \frac{\kappa}{2} \| w - x \|^2 \right\},
   \]

2. Estimate the gradient \( \nabla F(x) \)
   \[
   g = \kappa (x - z).
   \]

**output** approximate gradient estimate \( g \), objective value \( F_a \triangleq h(z) \), proximal mapping \( z \).
**Algorithm** QuickeNing

**input** $x_0$ in $\mathbb{R}^p$; number of iterations $K$; $\kappa > 0$; minimization algorithm $\mathcal{M}$.

1. **Initialization:** $(g_0, F_0, z_0) = \text{ApproxGradient}(x_0, \mathcal{M})$; $B_0 = \kappa I$.
2. **for** $k = 0, \ldots, K - 1$ **do**
   3. Perform the Quasi-Newton step
      \[
      x_{\text{test}} = x_k - B_k^{-1} g_k
      \]
      \[
      (g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \text{ApproxGradient}(x_{\text{test}}, \mathcal{M}).
      \]
   4. **if** $F_{\text{test}} \leq F_k - \frac{1}{2\kappa} ||g_k||^2$, **then**
   5. \[
   (x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}}).
   \]
   6. **else**
   7. Update the current iterate with the last proximal mapping:
      \[
      x_{k+1} = z_k = x_k - \frac{1}{\kappa} g_k
      \]
      \[
      (g_{k+1}, F_{k+1}, z_{k+1}) = \text{ApproxGradient}(x_{k+1}, \mathcal{M}).
      \]
   8. **end if**
   9. update $B_{k+1} = \text{L-BFGS}(B_k, x_{k+1} - x_k, g_{k+1} - g_k)$.
10. **end for**
**output** last proximal mapping $z_K$ (solution).
Algorithm QuickeNing

input $x_0$ in $\mathbb{R}^p$; number of iterations $K$; $\kappa > 0$; minimization algorithm $\mathcal{M}$.

1: Initialization: $(g_0, F_0, z_0) = \text{ApproxGradient}(x_0, \mathcal{M}); B_0 = \kappa I$.
2: for $k = 0, \ldots, K - 1$ do
3: Perform the Quasi-Newton step
4: \hspace{1em} $x_{\text{test}} = x_k - B_k^{-1} g_k$
5: \hspace{1em} $(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \text{ApproxGradient}(x_{\text{test}}, \mathcal{M})$.
6: if $F_{\text{test}} \leq F_{k-1}^2 \kappa \|g_k\|^2$ then
7: \hspace{1em} $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.
8: else
9: \hspace{1em} Update the current iterate with the last proximal mapping:
10: \hspace{2em} $x_{k+1} = z_k = x_k - \frac{1}{\kappa} g_k$.
11: Update $B_{k+1} = \text{L-BFGS}(B_k, x_{k+1} - x_k, g_{k+1} - g_k)$.
12: end if
13: end for

output last proximal mapping $z_K$ (solution).

The main characters:
- the sequence $(x_k)_{k \geq 0}$ that minimizes $F$;
- the sequence $(z_k)_{k \geq 0}$ produced by $\mathcal{M}$ that minimizes $f$;
- the gradient approximations $g_k \approx \nabla F(x_k)$;
- the function value approximations $F_k \approx F(x_k)$;
- an L-BFGS update with inexact gradients;
- an approximate sufficient descent condition.
Requirements on $\mathcal{M}$ and restarts

Method $\mathcal{M}$

Say a sub-problem consists of minimizing $h$; we want $\mathcal{M}$ to produce a sequence of iterates $(w_t)_{t \geq 0}$ with **linear convergence rate**

$$h(w_t) - h^* \leq C_\mathcal{M}(1 - \tau_\mathcal{M})^t(h(w_0) - h^*).$$

**Restarts**

- When $f$ is smooth, we **initialize** $w_0 = x$ when solving

$$\min_{w \in \mathbb{R}^d} \left\{ f(w) + \frac{\kappa}{2} \| w - x \|^2 \right\}.$$

- When $f = f_0 + \psi$ is composite, we use the initialization

$$w_0 = \arg \min_{w \in \mathbb{R}^d} \left\{ f_0(x) + \langle \nabla f_0(x), w - x \rangle + \frac{L + \kappa}{2} \| w - x \|^2 + \psi(w) \right\}.$$
When do we stop the method $\mathcal{M}$?

Three strategies to balance outer and inner computations

(a) use a **pre-defined sequence** $(\varepsilon_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the approximate proximal mapping is $\varepsilon_k$-accurate.

(b) define an **adaptive stopping criterion** that depends on quantities that are available at iteration $k$.

(c) use a **pre-defined budget** $T_{\mathcal{M}}$ of iterations of the method $\mathcal{M}$ for solving each sub-problem.
When do we stop the method $\mathcal{M}$?

Three strategies to balance outer and inner computations

(a) use a **pre-defined sequence** $(\varepsilon_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the approximate proximal mapping is $\varepsilon_k$-accurate.

(b) define an **adaptive stopping criterion** that depends on quantities that are available at iteration $k$.

(c) use a **pre-defined budget** $T_{\mathcal{M}}$ of iterations of the method $\mathcal{M}$ for solving each sub-problem.

Remarks

- We have already seen all of this for Catalyst.
When do we stop the method $\mathcal{M}$?

Three strategies for $\mu$-strongly convex objectives $f$

(a) use a pre-defined sequence $(\varepsilon_k)_{k \geq 0}$ and stop the optimization method $\mathcal{M}$ when the approximate proximal mapping is $\varepsilon_k$-accurate.

$$
\varepsilon_k = \frac{1}{2} C (1 - \rho)^{k+1} \quad \text{with} \quad C \geq f(x_0) - f^\star \quad \text{and} \quad \rho = \frac{\mu}{4(\mu + \kappa)}.
$$

(b) For minimizing $h(w) = f(w) + (\kappa/2)\|w - x\|^2$, stop when

$$
h(w_t) - h^\star \leq \frac{\kappa}{36} \|w_t - x\|^2.
$$

(c) use a pre-defined budget $T_M$ of iterations of the method $\mathcal{M}$ for solving each sub-problem with

$$
T_M = \frac{1}{\tau_M} \log \left( 19 C_M \frac{L + \kappa}{\kappa} \right). \quad \text{(be more aggressive in practice)}
$$
Remarks and worst-case global complexity

Composite objectives and sparsity

Consider a composite problem with a sparse solution (e.g., $\psi = \ell_1$). The method produces two sequences $(x_k)_{k \geq 0}$ and $(z_k)_{k \geq 0}$:

- $F(x_k) \to F^*$, minimizes the **smoothed objective** $\Rightarrow$ no sparsity;
- $f(z_k) \to f^*$, minimizes the **true objective** $\Rightarrow$ the iterates may be sparse if $\mathcal{M}$ handles composite optimization problems;

Global complexity

The number of iterations of $\mathcal{M}$ to guarantee $f(z_k) - f^* \leq \varepsilon$ is at most

- $\tilde{O}\left(\frac{\mu + \kappa}{\tau M \mu} \log(1/\varepsilon)\right)$ for $\mu$-strongly convex problems.
- $\tilde{O}\left(\frac{\kappa R^2}{\tau M \varepsilon}\right)$ for convex problems.
Global Complexity and choice of $\kappa$

Example for gradient descent

With the right step-size, we have $\tau_M = (\mu + \kappa)/(L + \kappa)$ and the complexity for $\mu > 0$ becomes

$$\tilde{O} \left( \frac{L + \kappa}{\mu} \log(1/\varepsilon) \right).$$

Example for SVRG for minimizing the sum of $n$ functions

$\tau_M = \min(1/n, (\mu + \kappa)/(L + \kappa))$ and the complexity for $\mu > 0$ is

$$\tilde{O} \left( \max \left( \frac{\mu + \kappa}{\mu} n, \frac{L + \kappa}{\mu} \right) \log(1/\varepsilon) \right).$$
Global Complexity and choice of $\kappa$

Example for gradient descent

With the right step-size, we have $\tau_M = (\mu + \kappa)/(L + \kappa)$ and the complexity for $\mu > 0$ becomes

$$\tilde{O} \left( \frac{L + \kappa}{\mu} \log(1/\varepsilon) \right).$$

Example for SVRG for minimizing the sum of $n$ functions

$\tau_M = \min(1/n, (\mu + \kappa)/(L + \kappa))$ and the complexity for $\mu > 0$ is

$$\tilde{O} \left( \max \left( \frac{\mu + \kappa}{\mu} n, \frac{L + \kappa}{\mu} \right) \log(1/\varepsilon) \right).$$

QuickeNing does not provide any theoretical acceleration, but it does not degrade significantly the worst-case performance of $M$ (unlike L-BFGS vs gradient descent).
Global Complexity and choice of $\kappa$

Example for gradient descent

With the right step-size, we have $\tau_M = (\mu + \kappa)/(L + \kappa)$ and the complexity for $\mu > 0$ becomes

$$\tilde{O} \left( \frac{L + \kappa}{\mu} \log\left(\frac{1}{\varepsilon}\right) \right).$$

Example for SVRG for minimizing the sum of $n$ functions

$\tau_M = \min(1/n, (\mu + \kappa)/(L + \kappa))$ and the complexity for $\mu > 0$ is

$$\tilde{O} \left( \max \left( \frac{\mu + \kappa}{\mu} n, \frac{L + \kappa}{\mu} \right) \log\left(\frac{1}{\varepsilon}\right) \right).$$

Then, how to choose $\kappa$?

(i) assume that L-BFGS steps do as well as Nesterov.

(ii) choose $\kappa$ as in Catalyst.
Experiments: formulations

- \( \ell_2 \)-regularized Logistic Regression:

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + \exp(-b_i a_i^T x) \right) + \frac{\mu}{2} \|x\|_2^2,
\]

- \( \ell_1 \)-regularized Linear Regression (LASSO):

\[
\min_{x \in \mathbb{R}^d} \frac{1}{2n} \sum_{i=1}^{n} (b_i - a_i^T x)^2 + \lambda \|x\|_1,
\]

- \( \ell_1 - \ell_2^2 \)-regularized Linear Regression (Elastic-Net):

\[
\min_{x \in \mathbb{R}^d} \frac{1}{2n} \sum_{i=1}^{n} (b_i - a_i^T x)^2 + \lambda \|x\|_1 + \frac{\mu}{2} \|x\|_2^2,
\]
Experiments: Datasets

We consider four standard machine learning datasets with different characteristics in terms of size and dimension:

<table>
<thead>
<tr>
<th>name</th>
<th>covtype</th>
<th>alpha</th>
<th>real-sim</th>
<th>rcv1</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>581 012</td>
<td>250 000</td>
<td>72 309</td>
<td>781 265</td>
</tr>
<tr>
<td>d</td>
<td>54</td>
<td>500</td>
<td>20 958</td>
<td>47 152</td>
</tr>
</tbody>
</table>

- we simulate the ill-conditioned regime $\mu = 1/(100n)$;
- $\lambda$ for the Lasso leads to about 10% non-zero coefficients.
Experiments: QuickeNing-SVRG

We consider the methods

- **SVRG**: the Prox-SVRG algorithm of Xiao and Zhang [2014].
- **Catalyst-SVRG**: Catalyst applied to SVRG;
- **L-BFGS** (for smooth objectives): Mark Schmidt’s implementation.
- **QuickeNing-SVRG1**: QuickeNing with aggressive strategy (c): one pass over the data in the inner loop.
- **QuickeNing-SVRG2**: strategy (b), compatible with theory.

We produce 12 figures (3 formulations, 4 datasets).
Experiments: QuickeNing-SVRG (log scale)

- covtype, logistic, $\mu = 1/100 \ n$
- covtype, lasso, $\lambda = 10 / n$
- rcv1, logistic, $\mu = 1/100 \ n$
- rcv1, lasso, $\lambda = 10 / n$

- QuickeNing-SVRG1 $\geq$ SVRG, QuickeNing-SVRG2;
- QuickeNing-SVRG2 $\geq$ SVRG;
- QuickeNing-SVRG1 $\geq$ Catalyst-SVRG in 10/12 cases.
Experiments: QuickeNing-ISTA

We consider the methods

- **ISTA**: the proximal gradient descent method with line search.
- **FISTA**: the accelerated ISTA of Beck and Teboulle [2009b].
- **L-BFGS** (for smooth objectives): Mark Schmidt’s implementation.
- **QuickeNing-ISTA1**: QuickeNing with aggressive strategy (c): one pass over the data in the inner loop.
- **QuickeNing-ISTA2**: strategy (b), compatible with theory.
Experiments: QuickeNing-ISTA (log scale)

- L-BFGS (for smooth $f$) is slightly better than QuickeNing-ISTA1;
- QuickeNing-ISTA $\geq$ or $\gg$ FISTA in 11/12 cases.
- QuickeNing-ISTA1 $\geq$ QuickeNing-ISTA2.
Experiments: Influence of $\kappa$

- $\kappa_0$ is the parameter (same as in Catalyst) used in all experiments;
- QuickeNing slows down when using $\kappa > \kappa_0$;
- here, for SVRG, QuickeNing is robust to small values of $\kappa$!
Experiments: Influence of $l$

- $l = 100$ in all previous experiments;
- $l = 5$ seems to be a reasonable choice in many cases, especially for sparse problems.
Conclusions and perspectives

- A simple generic Quasi-Newton method for composite functions, with simple sub-problems, and complexity guarantees.
- We also have a variant for dual approaches.
- Does not solve the gap between theory and practice for L-BFGS.

Perspectives

- QuickeNing-BCD, QuickeNing-SAG, SAGA, SDCA...
- Other types of smoothing? ⇒ Links with recent Quasi-Newton methods applied to other envelopes [Stella et al., 2016].
- Simple line search improves slightly the performance.
Outer-loop convergence analysis

Lemma: approximate descent property

\[ F(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{1}{4\kappa} \| \nabla F(x_k) \|_2^2 + 2\varepsilon_k. \]

Then, \( \varepsilon_k \) should be smaller than \( \frac{1}{4\kappa} \| \nabla F(x_k) \|_2^2 \), and indeed
Outer-loop convergence analysis

Lemma: approximate descent property

\[ F(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{1}{4\kappa} \| \nabla F(x_k) \|^2_2 + 2\varepsilon_k. \]

Then, \( \varepsilon_k \) should be smaller than \( \frac{1}{4\kappa} \| \nabla F(x_k) \|^2_2 \), and indeed

Proposition: convergence with impractical \( \varepsilon_k \) and \( \mu > 0 \)

If \( \varepsilon_k \leq \frac{1}{16\kappa} \| \nabla F(x_k) \|^2_2 \), define \( \rho = \frac{\mu}{4(\mu + \kappa)} \), then

\[ F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq (1 - \rho)^{k+1} (f(x_0) - f^*). \]

Unfortunately, \( \| \nabla F(x_k) \| \) is unknown.
Outer-loop convergence analysis

Lemma: approximate descent property

\[ F(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{1}{4\kappa} \|\nabla F(x_k)\|_2^2 + 2\varepsilon_k. \]

Then, \(\varepsilon_k\) should be smaller than \(\frac{1}{4\kappa}\|\nabla F(x_k)\|_2^2\), and indeed

Proposition: convergence with impractical \(\varepsilon_k\) and \(\mu > 0\)

If \(\varepsilon_k \leq \frac{1}{16\kappa} \|\nabla F(x_k)\|_2^2\), define \(\rho = \frac{\mu}{4(\mu + \kappa)}\), then

\[ F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq (1 - \rho)^{k+1} (f(x_0) - f^*). \]

Unfortunately, \(\|\nabla F(x_k)\|\) is unknown.

Lemma: convergence with adaptive \(\varepsilon_k\) and \(\mu > 0\)

If \(\varepsilon_k \leq \frac{1}{36\kappa} \|g_k\|_2^2\), then \(\varepsilon_k \leq \frac{1}{16} \|\nabla F(x_k)\|_2^2\).

This is strategy (b). \(g_k\) is known and easy to compute.
Inner-loop complexity analysis

Restart for $L$-smooth functions

For minimizing $h$, initialize the method $M$ with $w_0 = x$. Then,

$$h(w_0) - h^* \leq \frac{L + \kappa}{2\kappa^2} \|\nabla F(x)\|^2. \quad (1)$$

Proof.

We have the optimality condition $\nabla f(w^*) + \kappa (w^* - x) = 0$. As a result,

$$h(w_0) - h^*$$

$$= f(x) - \left( f(w^*) + \frac{\kappa}{2} \|w^* - x\|^2 \right)$$

$$\leq f(w^*) + \langle \nabla f(w^*), x - w^* \rangle + \frac{L}{2} \|x - w^*\|^2 - \left( f(w^*) + \frac{\kappa}{2} \|w^* - x\|^2 \right)$$

$$= \frac{L + \kappa}{2} \|w^* - x\|^2 = \frac{L + \kappa}{2\kappa^2} \|\nabla F(x)\|^2.$$


References II


References III


References V


References VI


