A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization

Julien Mairal

Inria, Grenoble

GdR Isis meeting at Telecom ParisTech

Collaborators

Hongzhou
Lin

Zaid
Harchaoui

Publications

H. Lin, J. Mairal and Z. Harchaoui. QuickeNing: A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization. *arXiv:1610.00960.* 2016

H. Lin, J. Mairal and Z. Harchaoui. A Universal Catalyst for First-Order Optimization. *Adv. NIPS* 2015.

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 ψ is a convex but not necessarily differentiable penalty, e.g., the ℓ_1-norm.

Goal of this work

- Design accelerated methods for minimizing **large finite sums**.
- Give **generic acceleration schemes** which can be applied 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.
- ψ 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 $\text{sign}(\langle z_i, x \rangle)$. The functions f_i measure how far the prediction is from the true label.

This would be a **classification problem with a linear model**.

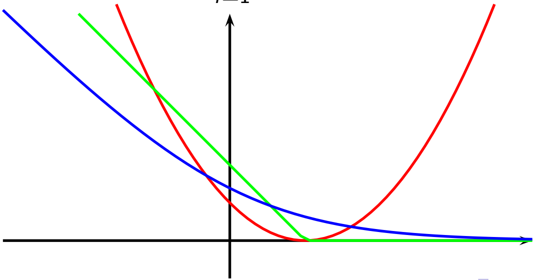
Why large finite sums matter?

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 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 ℓ_1-norm.

[Chen et al., 1999, Tibshirani, 1996].

Part I: a quick overview of optimization methods

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 μ-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

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

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

The classical forward-backward/ISTA algorithm

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

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

[Nowak and Figueiredo, 2001, Daubechies et al., 2004, Combettes and Wajs, 2006, Beck and Teboulle, 2009, Wright et al., 2009, Nesterov, 2013]...

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]

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

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

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

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

$$x_k \leftarrow x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1})$$

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

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

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

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

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

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

Main features vs. batch

- **Complexity per-iteration is n times smaller**;
- Convergence rate is slower: $O(1/k)$ 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

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

Main features vs. batch

- **Complexity per-iteration is n times smaller**;
- Convergence rate is slower: $O(1/k)$ for strongly-convex problems and $O(1/\sqrt{k})$ for convex ones, see [Nemirovski et al., 2009];
- variants are **compatible with prox** ψ, 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

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

Main features vs. batch

- **Complexity per-iteration is n times smaller**;
- Convergence rate is slower: $O(1/k)$ for strongly-convex problems and $O(1/\sqrt{k})$ for convex ones, see [Nemirovski et al., 2009];
- variants are **compatible with prox** ψ, 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.

Stochastic gradient descent methods

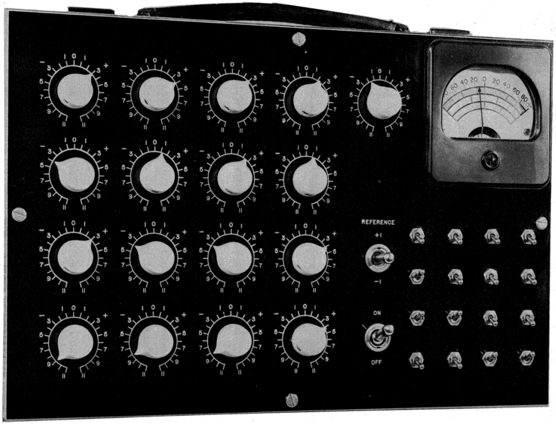


Figure: The Adaline [Widrow et al., 1960].

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 ∇f_i computed per iteration, with **fast convergence rates**, e.g., SAG [Schmidt et al., 2013]:

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

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 ∇f_i computed per iteration, with **fast convergence rates**, e.g., SAG [Schmidt et al., 2013]:

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

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., 2013, 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 $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

Incremental gradient descent methods

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

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

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 $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$

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 $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

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

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

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** ψ.

Incremental gradient descent methods

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

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

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** ψ.
- **May be accelerated [Lin, Mairal, and Harchaoui, 2015]**.

Yet, none of these approaches are able to exploit curvature.

Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Consider minimizing a twice-differentiable function $f(x)$.
- Newton-like methods use a quadratic approximation of f:

$$f(x_{k-1}) + \nabla f(x_{k-1})^\top (x - x_{k-1}) + \frac{1}{2\alpha}(x - x_{k-1}) B_k (x - x_{k-1}).$$

- B_k is a **positive-definite** approximation of the Hessian.
- The new iterate is set to the **minimizer of the approximation**

$$x_k \leftarrow x_{k-1} - \alpha d_k,$$

where d_k is the solution to

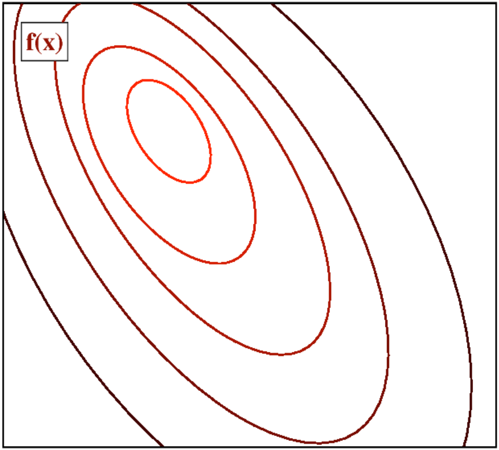
$$B_k d_k = \nabla f(x_{k-1}).$$

- Guarantees descent for small enough α.

Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

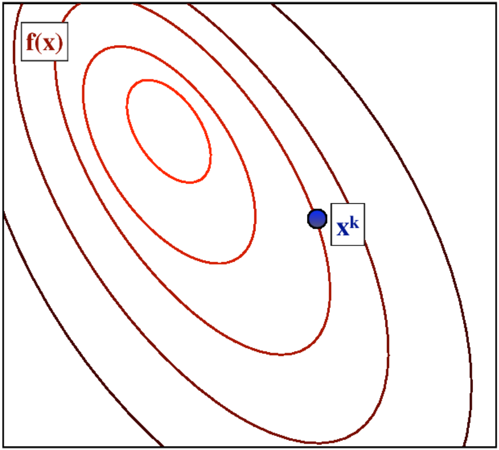
Newton-like vs gradient method.



Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

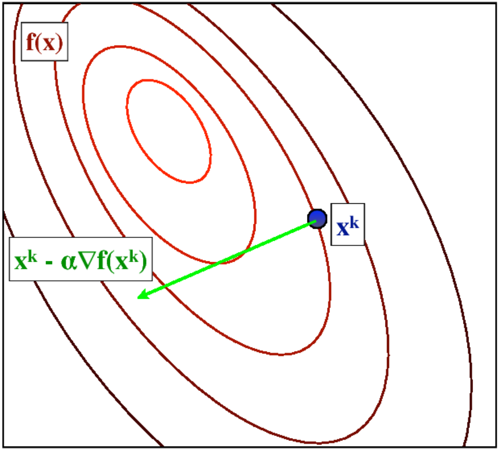
Newton-like vs gradient method.



Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

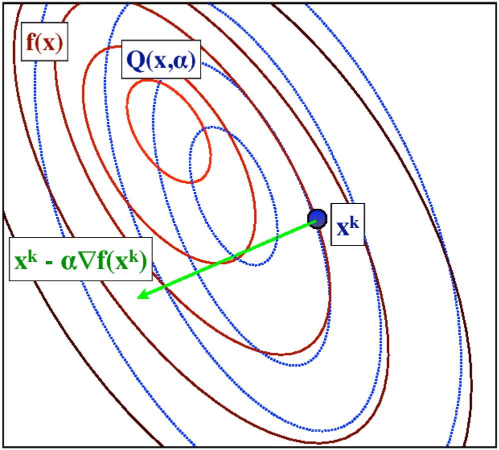
Newton-like vs gradient method.



Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

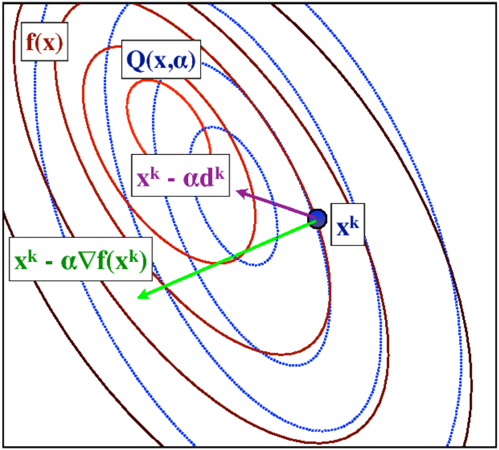
Newton-like vs gradient method.



Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

Newton-like vs gradient method.



Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

Pros

- Under suitable smoothness and convexity assumptions, the method achieves a **quadratic convergence rate**: it requires $O(\log \log 1/\varepsilon)$ iterations to achieve ε-accuracy.

Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

Pros

- Under suitable smoothness and convexity assumptions, the method achieves a **quadratic convergence rate**: it requires $O(\log\log 1/\varepsilon)$ iterations to achieve ε-accuracy.

Cons

- not always possible to **store and compute the $p \times p$ Hessian**...
- ... and even less possible to solve efficiently the linear systems.
- not clear how to deal efficiently with a **composite term**.

Newton-like methods

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

Pros

- Under suitable smoothness and convexity assumptions, the method achieves a **quadratic convergence rate**: it requires $O(\log \log 1/\varepsilon)$ iterations to achieve ε-accuracy.

Cons

- not always possible to **store and compute the $p \times p$ Hessian**...
- ... and even less possible to solve efficiently the linear systems.
- not clear how to deal efficiently with a **composite term**.

Alternatives

- solving inexactly the linear systems.
- **Limited Memory Quasi-Newton** (e.g., L-BFGS).

Broyden-Fletcher-Goldfarb-Shanno Quasi-Newton Method

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- **Quasi-Newton** methods work with the parameter and gradient differences between successive iterations:

$$s_k \triangleq x_{k+1} - x_k, \quad y_k \triangleq \nabla f(x_{k+1}) - \nabla f(x_k).$$

Broyden-Fletcher-Goldfarb-Shanno Quasi-Newton Method

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- **Quasi-Newton** methods work with the parameter and gradient differences between successive iterations:

$$s_k \triangleq x_{k+1} - x_k, \quad y_k \triangleq \nabla f(x_{k+1}) - \nabla f(x_k).$$

- They start with an initial approximation $B_0 \triangleq \sigma I$, and choose B_{k+1} to **interpolate the gradient difference**:

$$B_{k+1} s_k = y_k.$$

Broyden-Fletcher-Goldfarb-Shanno Quasi-Newton Method

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- **Quasi-Newton** methods work with the parameter and gradient differences between successive iterations:

$$s_k \triangleq x_{k+1} - x_k, \quad y_k \triangleq \nabla f(x_{k+1}) - \nabla f(x_k).$$

- They start with an initial approximation $B_0 \triangleq \sigma I$, and choose B_{k+1} to **interpolate the gradient difference**:

$$B_{k+1} s_k = y_k.$$

- Since B_{k+1} is not unique; the **BFGS** method chooses the symmetric matrix whose difference with B_k is minimal:

$$B_{k+1} = B_k - \frac{B_k s_k s_k B_k}{s_k B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}.$$

Convergence and Limited-Memory BFGS (L-BFGS)

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Update skipping/damping or a sophisticated line search (Wolfe conditions) can keep B_{k+1} positive-definite.

Convergence and Limited-Memory BFGS (L-BFGS)

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Update skipping/damping or a sophisticated line search (Wolfe conditions) can keep B_{k+1} positive-definite.
- The BFGS method has a **superlinear convergence rate**.

Convergence and Limited-Memory BFGS (L-BFGS)

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Update skipping/damping or a sophisticated line search (Wolfe conditions) can keep B_{k+1} positive-definite.
- The BFGS method has a **superlinear convergence rate**.
- But, it still uses a dense $p \times p$ matrix B_k.

Convergence and Limited-Memory BFGS (L-BFGS)

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Update skipping/damping or a sophisticated line search (Wolfe conditions) can keep B_{k+1} positive-definite.
- The BFGS method has a **superlinear convergence rate**.
- But, it still uses a dense $p \times p$ matrix B_k.
- Instead of storing B_k, the **limited-memory BFGS** (L-BFGS) method stores the previous l differences s_k and y_k.

Convergence and Limited-Memory BFGS (L-BFGS)

Presentation borrowed from Mark Schmidt, NIPS OPT 2010

- Update skipping/damping or a sophisticated line search (Wolfe conditions) can keep B_{k+1} positive-definite.
- The BFGS method has a **superlinear convergence rate**.
- But, it still uses a dense $p \times p$ matrix B_k.
- Instead of storing B_k, the **limited-memory BFGS** (L-BFGS) method stores the previous l differences s_k and y_k.
- We can solve a linear system involving these updates applied to a diagonal B_0 in $\mathcal{O}(pl)$ [Nocedal, 1980].

Limited-Memory BFGS (L-BFGS)

Remarks

- using the right initialization B_0 is crucial.
- the calibration of the line-search is also an art.

Limited-Memory BFGS (L-BFGS)

Remarks

- using the right initialization B_0 is crucial.
- the calibration of the line-search is also an art.

Pros

- **one of the biggest practical success of smooth optimization**.

Limited-Memory BFGS (L-BFGS)

Remarks

- using the right initialization B_0 is crucial.
- the calibration of the line-search is also an art.

Pros

- **one of the biggest practical success of smooth optimization**.

Cons

- worst-case convergence rates for strongly-convex functions are linear, but **no better than the gradient descent method**.
- proximal variants typically requires solving many times

$$\min_{x \in \mathbb{R}^p} \frac{1}{2}(x - z)B_k(z - z) + \psi(x).$$

- no guarantee of approximating the Hessian.

Part II: QuickeNing

Challenges

We still consider the problem

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

The goal is to

- **accelerate first-order methods** with Quasi-Newton principles.
- design L-BFGS algorithms **compatible with composite term**,
- which are **easy to use** (no line search, natural initialization, assuming L, μ are known),
- and which may **exploit the finite-sum structure**.

The workhorse: the Moreau-Yosida regularization

The Moreau-Yosida regularization of a convex function f is defined as

$$F(x) = \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|x - z\|^2 \right\},$$

and call $p(x)$ the unique solution of the problem.

The equivalence property

F is convex and minimizing f and F are equivalent in the sense that

$$\min_{x \in \mathbb{R}^p} F(x) = \min_{x \in \mathbb{R}^p} f(x).$$

The **minimizers of f and F coincide with each other**.

The workhorse: the Moreau-Yosida regularization

The Moreau-Yosida regularization of a convex function f is defined as

$$F(x) = \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|x - z\|^2 \right\},$$

and call $p(x)$ the unique solution of the problem.

The smoothness properties

- F is **continuously differentiable** even when f is not and

  $$\nabla F(x) = \kappa(x - p(x)),$$

  The gradient ∇F is Lipschitz continuous with constant $L_F = \kappa$.
- When f is μ-strongly convex, F is μ_F-strongly convex with constant $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$.
- $\Rightarrow$ When $\mu > 0$, the condition number of F is $1 + \frac{\kappa}{\mu}$.

The workhorse: the Moreau-Yosida regularization

A naive approach consists of minimizing 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

$$x_{k+1} = p(x_k) = \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2} \|z - x_k\|^2 \right\}.$$

This is exactly the **proximal point algorithm** [Rockafellar, 1976].

The workhorse: the Moreau-Yosida regularization

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 β_{k+1} is a Nesterov-like extrapolation parameter. We may now rewrite the update using the value of ∇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 workhorse: the Moreau-Yosida regularization

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 β_{k+1} is a Nesterov-like extrapolation parameter. We may now rewrite the update using the value of ∇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].

What is the advantage of these approaches?

F may be better conditioned than f when $1 + \kappa/\mu \leq L/\mu$;

The workhorse: the Moreau-Yosida regularization

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 β_{k+1} is a Nesterov-like extrapolation parameter. We may now rewrite the update using the value of ∇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].

What is the advantage of these approaches?

F may be better conditioned than f when $1 + \kappa/\mu \leq L/\mu$;

But...

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 approximately solving using an optimization method $\mathcal{M}$:

$$x_{k+1} \approx \operatorname*{arg\,min}_{x \in \mathbb{R}^p} \left\{ h_k(x) \triangleq f(x) + \frac{\kappa}{2}\|x - y_k\|^2 \right\},$$

such that $h_k(x_{k+1}) - h_k^* \leq \epsilon_k$.

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 approximately solving using an optimization method $\mathcal{M}$:

$$x_{k+1} \approx \arg\min_{x \in \mathbb{R}^p} \left\{ h_k(x) \triangleq f(x) + \frac{\kappa}{2}\|x - y_k\|^2 \right\},$$

such that $h_k(x_{k+1}) - h_k^* \leq \epsilon_k$.

Catalyst provides Nesterov's acceleration to $\mathcal{M}$ with...

- the right κ, sequence $(\varepsilon_k)_{k \geq 0}$, and restart strategy for $\mathcal{M}$.
- global complexity analysis resulting in theoretical acceleration.

QuickeNing

QuickeNing is a **limited memory Quasi-Newton algorithm with inexact gradients** applied to the smoothed function F.

Main features

- uses an optimization method $\mathcal{M}$ to solve the sub-problems.
- if $\mathcal{M}$ is compatible with prox, so is QuickeNing.
- **linear convergence rate for strongly-convex functions**.
- **no need for a line-search and easy initialization** of B_0, assuming L and μ are known.

QuickeNing

QuickeNing is a **limited memory Quasi-Newton algorithm with inexact gradients** applied to the smoothed function F.

Main features

- uses an optimization method $\mathcal{M}$ to solve the sub-problems.
- if $\mathcal{M}$ is compatible with prox, so is QuickeNing.
- **linear convergence rate for strongly-convex functions**.
- **no need for a line-search and easy initialization** of B_0, assuming L and μ are known.

Theory vs practice

- global theoretical complexity is not as good as Catalyst.
- in practice, outperforms Catalyst for ill-conditioned problems.

QuickeNing

Related work

- L-BFGS with inexact gradients [Friedlander and Schmidt, 2012].
- Quasi-Newton on Moreau-Yosida regularization [Burke and Qian, 2000, Chen and Fukushima, 1999, Fuentes et al., 2012, Fukushima and Qi, 1996].

Our contributions

- **practical** inexactness criterion and dedicated L-BFGS rule with no line search.
- **global complexity** with both inner- and outer-loop analysis.
- parameter choices that ensure **linear convergence rate for strongly-convex problems**.

The first building block

Algorithm Procedure GradientEstimate

input Current point x in $\mathbb{R}^p$; accuracy ε; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using $\mathcal{M}$:

$$z \approx \arg\min_{v \in \mathbb{R}^p} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v - x\|^2 \right\}, \tag{1}$$

such that $h(z) - h^* \leq \epsilon$ where $h^\star = \min_{z \in \mathbb{R}^p} h(z)$; define $F_a = h(z)$.

2: Estimate the gradient of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output gradient estimate $g \approx \nabla F(x)$,
objective value $F_a \approx F(x)$,
proximal mapping $z \approx p(x)$.

The first building block

Algorithm Procedure GradientEstimate

input Current point x in $\mathbb{R}^p$; accuracy ε; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using $\mathcal{M}$:

$$z \approx \arg\min_{v \in \mathbb{R}^p} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v - x\|^2 \right\}, \tag{1}$$

such that $h(z) - h^* \leq \epsilon$ where $h^\star = \min_{z \in \mathbb{R}^p} h(z)$; define $F_a = h(z)$.

2: Estimate the gradient of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output gradient estimate $g \approx \nabla F(x)$,
objective value $F_a \approx F(x)$,
proximal mapping $z \approx p(x)$.

The first building block

Algorithm Procedure GradientEstimate

input Current point x in $\mathbb{R}^p$; accuracy ε; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using $\mathcal{M}$:

$$z \approx \arg\min_{v \in \mathbb{R}^p} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v - x\|^2 \right\}, \tag{1}$$

such that $h(z) - h^* \leq \epsilon$ where $h^\star = \min_{z \in \mathbb{R}^p} h(z)$; define $F_a = h(z)$.

2: Estimate the gradient of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output gradient estimate $g \approx \nabla F(x)$,
objective value $F_a \approx F(x)$,
proximal mapping $z \approx p(x)$.

The first building block

Algorithm Procedure GradientEstimate

input Current point x in $\mathbb{R}^p$; accuracy ε; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using $\mathcal{M}$:

$$z \approx \arg\min_{v \in \mathbb{R}^p} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2} \|v - x\|^2 \right\}, \qquad (1)$$

such that $h(z) - h^* \leq \epsilon$ where $h^\star = \min_{z \in \mathbb{R}^p} h(z)$; define $F_a = h(z)$.

2: Estimate the gradient of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output gradient estimate $g \approx \nabla F(x)$,
objective value $F_a \approx F(x)$,
proximal mapping $z \approx p(x)$.

The first building block

Remember,

$$F(x) = \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|x - z\|^2 \right\},$$

and call $p(x)$ the unique solution of the problem.

Approximation guarantees [Fukushima and Qi, 1996]

Consider a vector x in $\mathbb{R}^p$, a positive scalar ε and

$$(g, F_a, z) = \text{GradientEstimate}(x, \varepsilon).$$

Then, the following inequalities hold

$$F(x) \leq F_a \leq F(x) + \varepsilon,$$

$$\|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}},$$

$$\|g - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}.$$

Second building block: dedicated L-BFGS rule

- Initialize $C_1 = (1/\kappa)I$.
- Maintain a generating list $(s_i, y_i)_{i=1\ldots j}$ with $j \leq l$ such that

$$C_{i+1} = C_i - \frac{C_i s_i s_i C_i}{s_i C_i s_i} + \frac{y_i y_i^\top}{y_i^\top s_i}$$

  and the current L-BFGS matrix is $B_k = C_j$.
- Remember that B_k is never stored explicitly, but that $B_k^{-1}z$ can be computed in $O(pl)$ operations for all vector z.
- The generating list is incrementally updated given a new pair

$$y_k \approx \nabla F(x_{k+1}) - \nabla F(x_k) \quad \text{and} \quad s_k = x_{k+1} - x_k.$$

  but it requires **skipping steps** to ensure positive definitess.

Second building block: dedicated L-BFGS rule

Algorithm Quasi-Newton-type update rule L-BFGS

input current generating list $\{(s_i, y_i)\}_{i=1...j}$; new candidate pair (s, y); L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

1: **if** the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s.$$

then

2: add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

3: **else**

4: keep the generating list unchanged.

5: **end if**

output new L-BFGS matrix B (generating list).

Second building block: dedicated L-BFGS rule

Algorithm Quasi-Newton-type update rule L-BFGS

input current generating list $\{(s_i, y_i)\}_{i=1...j}$; new candidate pair (s, y); L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

1: **if** the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s.$$

then

2: add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

3: **else**

4: keep the generating list unchanged.

5: **end if**

output new L-BFGS matrix B (generating list).

Second building block: dedicated L-BFGS rule

Algorithm Quasi-Newton-type update rule L-BFGS

input current generating list $\{(s_i, y_i)\}_{i=1...j}$; new candidate pair (s, y); L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

1: **if** the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s.$$

then

2: add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

3: **else**

4: keep the generating list unchanged.

5: **end if**

output new L-BFGS matrix B (generating list).

Second building block: dedicated L-BFGS rule

Algorithm Quasi-Newton-type update rule L-BFGS

input current generating list $\{(s_i, y_i)\}_{i=1...j}$; new candidate pair (s, y);
L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

if the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s.$$

then

add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

else

keep the generating list unchanged.

end if

output new L-BFGS matrix B (generating list).

Second building block: dedicated L-BFGS rule

Algorithm Quasi-Newton-type update rule L-BFGS

input current generating list $\{(s_i, y_i)\}_{i=1...j}$; new candidate pair (s, y);
L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

if the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s.$$

then

add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

else

keep the generating list unchanged.

end if

output new L-BFGS matrix B (generating list).

Finally, the QuickeNing algorithm I

Algorithm QuickeNing

input Initial point x_0 in $\mathbb{R}^p$; sequence $(\varepsilon_k)_{k\geq 0}$; number of iterations K; smoothing parameter $\kappa > 0$; L-BFGS parameters $0 < c_1, c_2 \leq 1$; optimization method $\mathcal{M}$.

1: **Initialization**:
$(g_0, F_0, z_0) = \text{GradientEstimate}(x_0, \varepsilon_0)$;
BFGS matrix $B_0 = \kappa I$.

2: **for** $k = 0, \ldots, K-1$ **do**

3: Perform the Quasi-Newton step

$$x_{\text{test}} = x_k - B_k^{-1} g_k.$$

4: Estimate the new gradient and the Moreau-Yosida function value

$$(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \text{GradientEstimate}(x_{\text{test}}, \varepsilon_{k+1}).$$

Finally, the QuickeNing algorithm II

if sufficient decrease is obtained

$$F_{\text{test}} \leq F_k - \frac{1}{4\kappa}\|g_k\|^2 + \epsilon_k, \tag{2}$$

then

accept: $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.

else

update the current iterate: $x_{k+1} = z_k$.

$$(g_{k+1}, F_{k+1}, z_{k+1}) = \text{GradientEstimate}(x_{k+1}, \varepsilon_{k+1}).$$

end if

update $B_{k+1} = \text{L-BFGS}(B_k, x_{k+1} - x_k, g_{k+1} - g_k)$.

end for

output last proximal mapping z_K (solution).

Convergence analysis

A key lemma:

Approximate descent property

Consider the sequence $(x_k, z_k)_{k \geq 0}$ generated by QuickeNing. Then,

$$\max\{F(x_{k+1}), f(z_k)\} \leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k.$$

Convergence analysis

A key lemma:

Approximate descent property

Consider the sequence $(x_k, z_k)_{k \geq 0}$ generated by QuickeNing. Then,

$$\max\{F(x_{k+1}), f(z_k)\} \leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k.$$

In contrast, the exact gradient descent method applied to F provides

$$F(x_{k+1}) \leq F(x_k) - \frac{1}{2\kappa}\|\nabla F(x_k)\|^2.$$

Convergence analysis

Next, we control the accumulation of errors.

Accumulation of errors in QuickeNing when $\mu > 0$

Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{8(\mu+\kappa)}$. Then, the iterates $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ produced by QuickeNing satisfy

$$\max\{F(x_{k+1}) - F^*, f(z_k) - f^*\} \leq \\ (1-2\rho)^{k+1}(f(x_0) - f^*) + 3\sum_{i=0}^{k}(1-2\rho)^{k-i}\varepsilon_i.$$

Convergence analysis

Complexity analysis when $\mu > 0$

Assume that $\mathcal{M}$ is always able to produce a sequence of iterates $(w_t)_{t\geq 0}$ for solving the sub-problems such that

$$h(w_t) - h^\star \leq A(1-\tau_{\mathcal{M}})^t (h(w_0) - h^\star) \quad \text{for some constants} \quad A, \tau_{\mathcal{M}} > 0. \tag{3}$$

Then, choose $\varepsilon_k = C(1-\rho)^{k+1}/3$ with $C \geq (f(x_0) - f^\star)$ and define $\rho = \frac{\mu}{8(\mu+\kappa)}$; then,

$$\max\left\{F(x_k) - F^\star, f(z_k) - f^\star\right\} \leq \frac{C}{\rho}(1-\rho)^{k+2}. \tag{4}$$

Moreover, by initializing $\mathcal{M}$ with $w_0 = z_k$ at iteration k, each sub-problem (1) is solved up to accuracy ε_{k+1} in at most a constant number $T_{\mathcal{M}}$ of iterations of $\mathcal{M}$, where $T_{\mathcal{M}} = \tilde{O}(1/\tau_{\mathcal{M}})$.

Remarks

Theory and practice

- the restart at z_k is not the best one, both in theory and in practice (work in progress, arXiv paper is outdated).
- the gap between theory and practice is huge, due to L-BFGS.
- the theory does not provide the right parameters for κ: we use those of Catalyst in practice.

Nice features

- $\mathcal{M}$ can **exploit the structure** (incremental for large n, block coordinate descent for large p), and so does QuickeNing.
- **no line search**: when the test point is rejected, we perform one step of inexact PPA, whose convergence is well understood.
- the sequence $(z_k)_{k \geq 0}$ is produced by $\mathcal{M}$ and thus may be **compatible with composite regularization** (e.g., sparse).

Part III: Preliminary experiments

Formulations

We consider two types of formulations

A smooth one: logistic regression

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

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

μ is the regularization parameter.

A non-smooth one: Elastic-net [Zou and Hastie, 2005]

$$\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - x^\top z_i)^2 + \lambda \|x\|_1 + \frac{\mu}{2} \|x\|_2^2.$$

We will consider a regime with relatively small μ.

Datasets and methods

Datasets

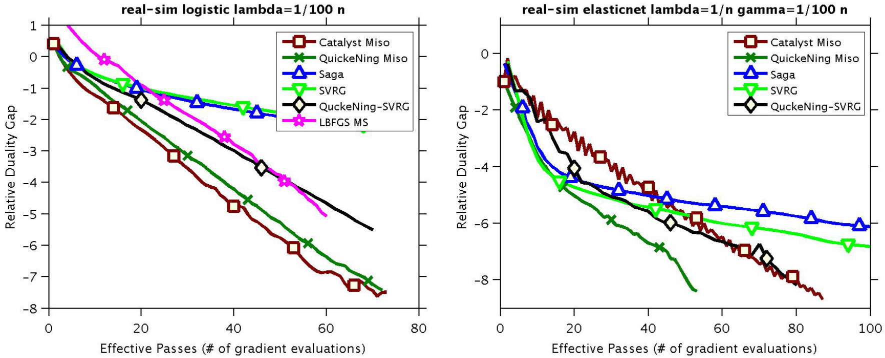
name	rcv1	real-sim	covtype	alpha
n	781 265	72 309	581 012	250 000
p	47 152	20 958	54	500

Methods

- Mark Schmidt's implementation of L-BFGS;
- Catalyst Miso [Lin, Mairal, and Harchaoui, 2015];
- QuickeNing Miso;
- SAGA [Defazio et al., 2014a];
- SVRG [Xiao and Zhang, 2014];
- QuickeNing SVRG.

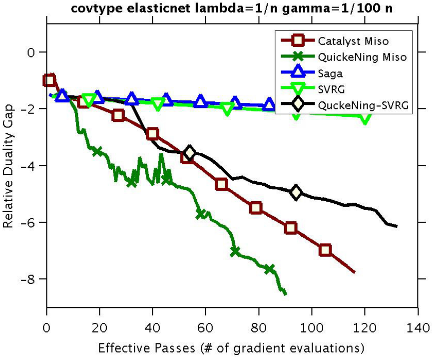
All methods come with **default parameters** (no further tuning here).

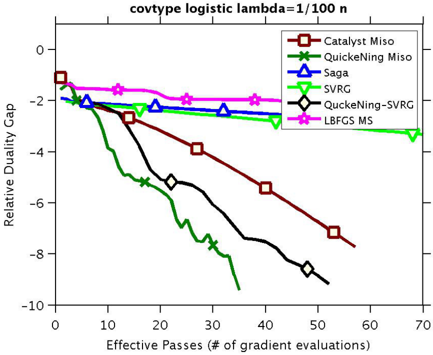
real-sim



- **QuickeNing MISO** $\geq$ Catalyst MISO.
- QuickeNing SVRG $>$ SVRG.
- L-BFGS is competitive, unlike SAGA.

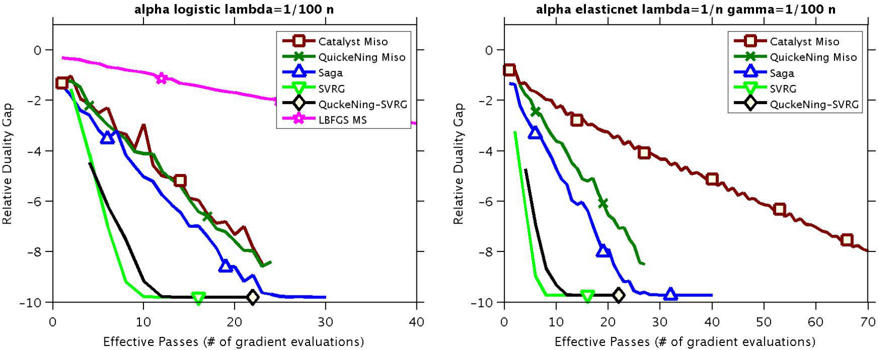
covtype





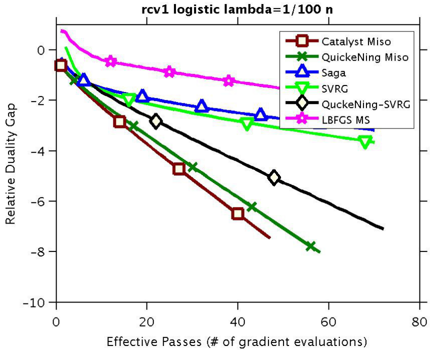
- **QuickeNing MISO** $\geq$ Catalyst MISO.
- QuickeNing SVRG $>$ SVRG.
- L-BFGS and SAGA are not competitive here.

alpha



- **QuickeNing SVRG** and **SVRG** are surprinsingly good.
- QuickeNing MISO $\geq$ Catalyst MISO.
- SAGA is close to QuickeNing MISO here.

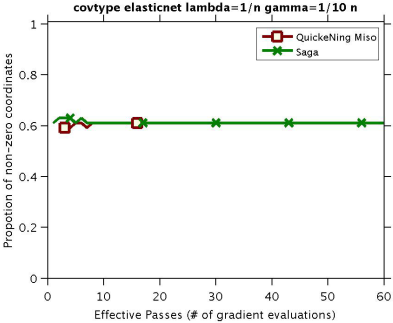
rcv1

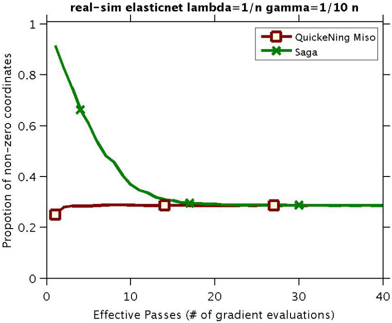


- **QuickeNing MISO** and **Catalyst MISO** are the best here.
- QuickeNing SVRG > SVRG.
- QuickeNing MISO $\geq$ Catalyst MISO.

QuickeNing and sparsity

Are the iterates (z_k) sparse with the Elastic-Net?





When the regularization parameter λ is large enough, the solution is sparse. In this context, **exact sparsity is a desirable feature**.

Concluding remarks

- **Conclusions are always data/context-dependent:**
  - Is the dataset well-conditioned?
  - What is the amount of regularization?
  - Is there hidden strong convexity in the loss at the optimum?
  - Is the solution sparse?
- **QuickeNing has been a safe heuristic so far**.
- Not evaluated yet: **the one-pass heuristic, QuickeNing-block-coordinate-descent, . . .** .
- We also have convergence results without strong convexity, but no complexity analysis.

Concluding remarks

- **Conclusions are always data/context-dependent:**
  - Is the dataset well-conditioned?
  - What is the amount of regularization?
  - Is there hidden strong convexity in the loss at the optimum?
  - Is the solution sparse?
- **QuickeNing has been a safe heuristic so far**.
- Not evaluated yet: **the one-pass heuristic, QuickeNing-block-coordinate-descent, . . .** .
- We also have convergence results without strong convexity, but no complexity analysis.
- Note: this is **work in progress**; the figures here should not be considered as those of a published paper (yet).

References I

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

J.V. Burke and Maijian Qian. On the superlinear convergence of the variable metric proximal point algorithm using Broyden and BFGS matrix secant updating. *Mathematical Programming*, 88(1):157–181, 2000.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.

Xiaojun Chen and Masao Fukushima. Proximal quasi-Newton methods for nondifferentiable convex optimization. *Mathematical Programming*, 85(2):313–334, 1999.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4):1168–1200, 2006.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

References II

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014a.

A. J. Defazio, T. S. Caetano, and J. Domke. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014b.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3): A1380–A1405, 2012.

Marc Fuentes, Jérôme Malick, and Claude Lemaréchal. Descentwise inexact proximal algorithms for smooth optimization. *Computational Optimization and Applications*, 53(3):755–769, 2012.

References III

Masao Fukushima and Liqun Qi. A globally and superlinearly convergent algorithm for nonsmooth convex minimization. *SIAM Journal on Optimization*, 6(4):1106–1120, 1996.

O. Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 2015.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2): 829–855, 2015.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Publishers, 2004.

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

References IV

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983.

R. D. Nowak and M. A. T. Figueiredo. Fast wavelet-based image deconvolution using the EM algorithm. In *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers.*, 2001.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

S. Shalev-Shwartz and T. Zhang. Proximal stochastic dual coordinate ascent. *arXiv:1211.2717*, 2012.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society Series B*, 58(1):267–288, 1996.

Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

References V

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

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.