On Accelerated Optimization Methods for Large-Scale Machine Learning

Julien Mairal

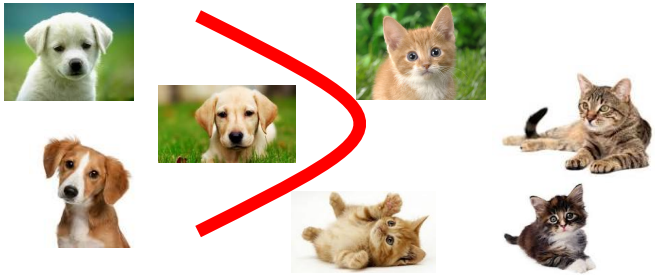
Inria Grenoble

ICT Innovations 2020, Skopje (online)

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, h(a_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$



Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, h(a_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$

The labels b_i are in

- $\{-1, +1\}$ for **binary** classification.
- $\{1, \dots, K\}$ for **multi-class** classification.
- $\mathbb{R}$ for **regression**.
- $\mathbb{R}^k$ for **multivariate regression**.
- any general set for **structured prediction**.

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, h(a_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$

The empirical risk minimization (ERM) paradigm

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

Very Popperian point of view, see [Vapnik, 1995, Corfield, Schölkopf, and Vapnik, 2009]...

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, h(a_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$

The empirical risk minimization (ERM) paradigm, parenthesis on limitations: "("

- it is not always possible to estimate the generalization error based on available data.
- when a complex model A performs slightly better than a simple model B, should we prefer A or B?
- we are also leaving aside the problem of non i.i.d. train/test data, biased data, testing with counterfactual reasoning... ")"

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{x \in \mathbb{R}^p} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, x^\top a_i)}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(x)}_{\text{regularization}} .$$

Example: linear models

- assume there exists a linear relation between b in $\mathbb{R}$ and features a in $\mathbb{R}^p$.
- $h(x) = a^\top x = \sum_j a[j]x[j]$ is parametrized by x in $\mathbb{R}^p$.
- L is often a **convex** loss function.
- Ω is often the squared ℓ_2-norm $\|x\|^2$, but the ℓ_1-norm is also very popular.

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{A} \to \mathcal{B}$ given labeled training data $(a_i, b_i)_{i=1,\dots,n}$ with a_i in $\mathcal{A}$, and b_i in $\mathcal{B}$:

$$\min_{x \in \mathbb{R}^p} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(b_i, x^\top a_i)}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(x)}_{\text{regularization}} .$$

Why the ℓ_2-regularization for linear models $h(a) = x^\top a$?

- Intuition: if a and a' are similar, so should $h(a)$ and $h(a')$ be:

$$|h(a) - h(a')| \leq \|x\|_2 \|a - a'\|_2.$$

- Because we have **theory** for it (and it works in practice)!

Optimization is central to machine learning

A few examples of linear models:

Ridge regression: $$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(b_i - x^\top a_i)^2 + \lambda\|x\|_2^2.$$

Linear SVM: $$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - b_i x^\top a_i) + \lambda\|x\|_2^2.$$

Logistic regression: $$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-b_i x^\top a_i}\right) + \lambda\|x\|_2^2.$$

Loss as a function of $x^\top a$ with $b = 1$.

What are we interested in?

Our goal is to learn linear models on regular workstations

- with potentially large datasets that fit into memory (e.g., ≤ 256Gb).
- with various loss (regression, classification) and regularization functions (ℓ_2, ℓ_1, ...).

What are we interested in?

Our goal is to learn linear models on regular workstations

- with potentially large datasets that fit into memory (e.g., ≤ 256Gb).
- with various loss (regression, classification) and regularization functions (ℓ_2, ℓ_1, ...).

Algorithms that are

- **fast** (exploit the structure of the problem).
- **robust to difficult problems** (numerically stable).
- **with optimization guarantees** (crucial for reproducibility).

What are we interested in?

Our goal is to learn linear models on regular workstations

- with potentially large datasets that fit into memory (e.g., ≤ 256Gb).
- with various loss (regression, classification) and regularization functions (ℓ_2, ℓ_1, ...).

Algorithms that are

- **fast** (exploit the structure of the problem).
- **robust to difficult problems** (numerically stable).
- **with optimization guarantees** (crucial for reproducibility).

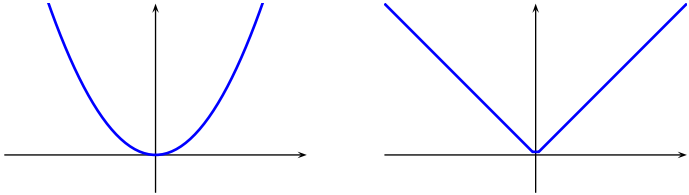
Software packages that are

- **memory-efficient** (no data copy).
- **resource-efficient** (exploit low-level languages and libraries, C++/BLAS).
- **easy to use** (scikit-learn compatible API, available in many high-level languages).

Part I: Algorithms and mathematical principles

Basics of gradient-based optimization

Smooth vs non-smooth



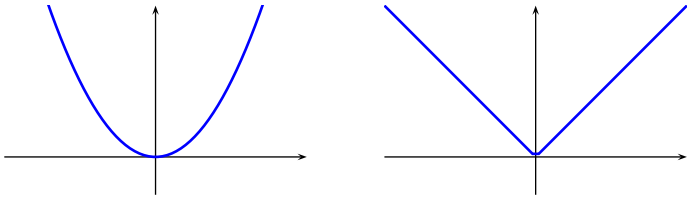
(a) smooth (b) non-smooth

An important quantity to quantify smoothness is the **Lipschitz constant** of the gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

Basics of gradient-based optimization

Smooth vs non-smooth



(a) smooth (b) non-smooth

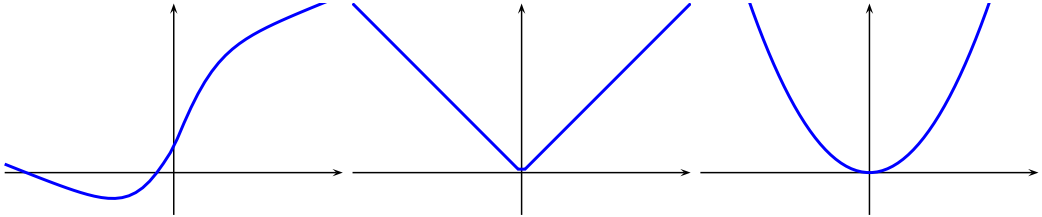
An important quantity to quantify smoothness is the **Lipschitz constant** of the gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

If f is twice differentiable, L may be chosen as the **largest eigenvalue** of the Hessian $\nabla^2 f$. This is an upper-bound on the function curvature.

Basics of gradient-based optimization

Convex vs non-convex



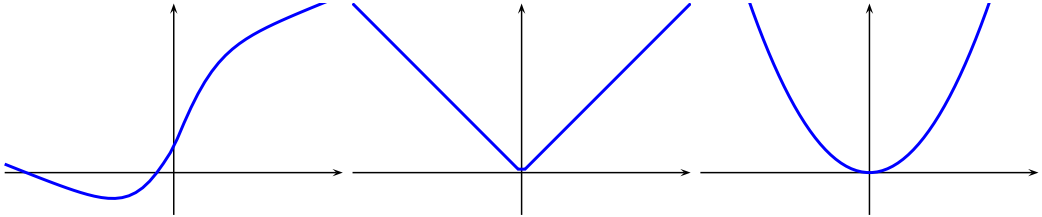
(a) non-convex (b) convex (c) strongly-convex

An important quantity to quantify convexity is the **strong-convexity** constant

$$f(x) \geq f(y) + \nabla f(y)^\top (x - y) + \frac{\mu}{2}\|x - y\|^2,$$

Basics of gradient-based optimization

Convex vs non-convex



(a) non-convex (b) convex (c) strongly-convex

An important quantity to quantify convexity is the **strong-convexity** constant

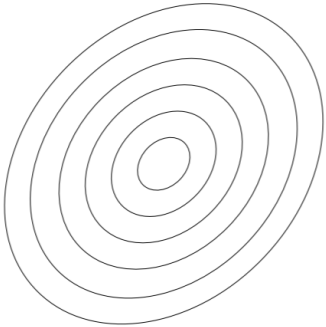
$$f(x) \geq f(y) + \nabla f(y)^\top (x - y) + \frac{\mu}{2}\|x - y\|^2,$$

If f is twice differentiable, μ may be chosen as the **smallest eigenvalue** of the Hessian $\nabla^2 f$. This is a lower-bound on the function curvature.

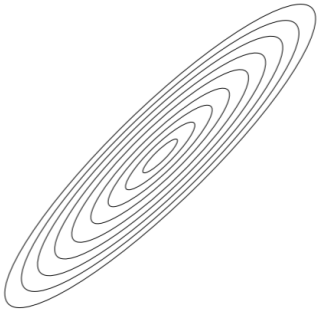
Basics of gradient-based optimization

Picture from F. Bach

Why is the condition number L/μ important?



(small $\kappa = L/\mu$)

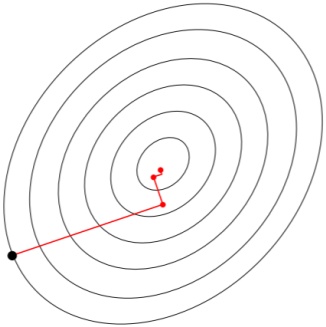


(large $\kappa = L/\mu$)

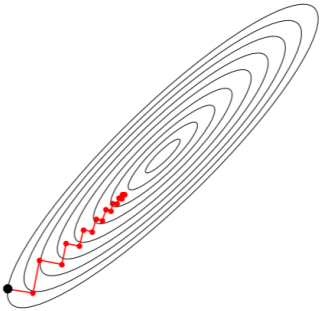
Basics of gradient-based optimization

Picture from F. Bach

Trajectory of gradient descent with optimal step size $x_t \leftarrow x_{t-1} - \eta_t \nabla f(x_{t-1})$.



(small $\kappa = L/\mu$)



(large $\kappa = L/\mu$)

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Proposition

If f is μ-strongly convex and differentiable with L-Lipschitz gradient, the gradient descent method finds an ε-solution in at most $O((L/\mu)\log(1/\varepsilon))$ iterations.

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Proposition

If f is μ-strongly convex and differentiable with L-Lipschitz gradient, the gradient descent method finds an ε-solution in at most $O((L/\mu)\log(1/\varepsilon))$ iterations.

Can we do better?

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Proposition

If f is μ-strongly convex and differentiable with L-Lipschitz gradient, the gradient descent method finds an ε-solution in at most $O((L/\mu)\log(1/\varepsilon))$ iterations.

Can we do better? Yes, Nesterov [1983] proposes a method with complexity $O(\sqrt{L/\mu}\log(1/\varepsilon))$

$$x_t \leftarrow y_{t-1} - \eta_t \nabla f(y_{t-1})$$
$$y_t \leftarrow x_t + \beta_t(x_t - x_{t-1})$$

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Proposition

If f is μ-strongly convex and differentiable with L-Lipschitz gradient, the gradient descent method finds an ε-solution in at most $O((L/\mu)\log(1/\varepsilon))$ iterations.

Can we do better? Yes, Nesterov [1983] proposes a method with complexity $O(\sqrt{L/\mu}\log(1/\varepsilon))$

$$x_t \leftarrow y_{t-1} - \eta_t \nabla f(y_{t-1})$$
$$y_t \leftarrow x_t + \beta_t(x_t - x_{t-1})$$

Can we do better?

Complexity and accelerated gradient descent (Idea 1)

A natural question is how many iterations are required to guarantee $f(x_t) - f^\star \leq \varepsilon$?

Proposition

If f is μ-strongly convex and differentiable with L-Lipschitz gradient, the gradient descent method finds an ε-solution in at most $O((L/\mu)\log(1/\varepsilon))$ iterations.

Can we do better? Yes, Nesterov [1983] proposes a method with complexity $O(\sqrt{L/\mu}\log(1/\varepsilon))$

$$x_t \leftarrow y_{t-1} - \eta_t \nabla f(y_{t-1})$$
$$y_t \leftarrow x_t + \beta_t(x_t - x_{t-1})$$

Can we do better? No, unless

- you consider problems with **a specific structure.**
- your algorithm is **not deterministic**.

Exploiting the structure with stochastic approximations (Idea 2)

The machine learning problems we consider are **large finite sums of functions**

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

Computing the gradient $\nabla F(x)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x)$ requires **computing n gradients** of the functions f_i. The complexity of gradient descent becomes $O((nL/\mu)\log(1/\varepsilon))$.

The stochastic gradient descent method [Robbins and Monro, 1951]

$$x_t \leftarrow x_{t-1}-\eta_t\nabla f_{i_t}(x_{t-1})$$

The complexity per iteration is $O(1)$ instead of $O(n)$, but we **lose the logarithmic dependency** in ε [see, e.g. Polyak and Juditsky, 1992]; with averaging, the typical complexity is $O(1/\mu\varepsilon)$ for strongly convex problems.

Stochastic approximations with variance reduction (Idea 3)

The stochastic gradient descent method uses an **unbiased estimate** of the gradient

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_{i_t}(x_{t-1}) \quad \text{such that} \quad \mathbb{E}[\nabla f_{i_t}(x_{t-1})] = \nabla F(x_{t-1}).$$

Stochastic approximations with variance reduction (Idea 3)

The stochastic gradient descent method uses an **unbiased estimate** of the gradient

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_{i_t}(x_{t-1}) \quad \text{such that} \quad \mathbb{E}[\nabla f_{i_t}(x_{t-1})] = \nabla F(x_{t-1}).$$

Can we find a better estimate of the gradient (leading to better complexity)?

Stochastic approximations with variance reduction (Idea 3)

The stochastic gradient descent method uses an **unbiased estimate** of the gradient

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_{i_t}(x_{t-1}) \quad \text{such that} \quad \mathbb{E}[\nabla f_{i_t}(x_{t-1})] = \nabla F(x_{t-1}).$$

Can we find a better estimate of the gradient (leading to better complexity)?
Consider two random variables X, Y and define

$$Z = X - Y + \mathbb{E}[Y].$$

Then,

- $\mathbb{E}[Z] = \mathbb{E}[X]$ and $\mathrm{Var}(Z) = \mathrm{Var}(X) + \mathrm{Var}(Y) - 2\mathrm{cov}(X, Y)$.

The variance of Z may be smaller if X and Y are positively correlated.

Stochastic approximations with variance reduction (Idea 3)

The stochastic gradient descent method uses an **unbiased estimate** of the gradient

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_{i_t}(x_{t-1}) \quad \text{such that} \quad \mathbb{E}[\nabla f_{i_t}(x_{t-1})] = \nabla F(x_{t-1}).$$

Can we find a better estimate of the gradient (leading to better complexity)?
Consider two random variables X, Y and define

$$Z = X - Y + \mathbb{E}[Y].$$

Then,

- $\mathbb{E}[Z] = \mathbb{E}[X]$ and $\mathsf{Var}(Z) = \mathsf{Var}(X) + \mathsf{Var}(Y) - 2\mathsf{cov}(X, Y)$.

The variance of Z may be smaller if X and Y are positively correlated.

The idea is now to **use past gradients** $\nabla f_{i_t}(x_{\mathsf{past}})$ to **reduce the variance** of $\nabla f_{i_t}(x_{t-1})$.

Stochastic approximations with variance reduction (Idea 3)

The stochastic gradient descent method uses an **unbiased estimate** of the gradient

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_{i_t}(x_{t-1}) \quad \text{such that} \quad \mathbb{E}[\nabla f_{i_t}(x_{t-1})] = \nabla F(x_{t-1}).$$

Can we find a better estimate of the gradient (leading to better complexity)?
Consider two random variables X, Y and define

$$Z = X - Y + \mathbb{E}[Y].$$

Then,

- $\mathbb{E}[Z] = \mathbb{E}[X]$ and $\mathsf{Var}(Z) = \mathsf{Var}(X) + \mathsf{Var}(Y) - 2\mathsf{cov}(X, Y)$.

The variance of Z may be smaller if X and Y are positively correlated.

The idea is now to **use past gradients** $\nabla f_{i_t}(x_{\mathsf{past}})$ to **reduce the variance** of $\nabla f_{i_t}(x_{t-1})$.

SAG [Schmidt et al., 2013], SVRG [Xiao and Zhang, 2014], SAGA [Defazio et al., 2014], SDCA [Shalev-Shwartz and Zhang, 2012], **MISO** [Mairal, 2015], and many others ...

Stochastic approximations with variance reduction (Idea 3)

These methods achieve low **(worst-case)** complexity in expectation. The number of gradients evaluations to ensure $\mathbb{E}[f(x_t) - f^\star] \leq \varepsilon$ is

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

Stochastic approximations with variance reduction (Idea 3)

These methods achieve low **(worst-case)** complexity in expectation. The number of gradients evaluations to ensure $\mathbb{E}[f(x_t) - f^\star] \leq \varepsilon$ is

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

Main features vs. stochastic gradient descent

- Same complexity per-iteration.
- **Faster convergence** (exploit the finite-sum structure).
- **Less parameter tuning** than SGD.
- SVRG is better than acc-GD if $n \geq \sqrt{L/\mu}$.

Stochastic approximations with variance reduction (Idea 3)

These methods achieve low **(worst-case)** complexity in expectation. The number of gradients evaluations to ensure $\mathbb{E}[f(x_t) - f^\star] \leq \varepsilon$ is

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

Can we do better? Yes, with acceleration: The method Katyusha [Allen-Zhu, 2016] (and others) achieve

$$O\left(\left(n + \sqrt{\frac{nL}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$$

Other ideas and recent contributions

Other important concepts

- **duality gaps**: convex optimization offers mechanisms to control $f(x_t) - f^\star$ in practice!
- **composite optimization**: all previous approaches can be extended to solve

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

  where Ω is convex and non-smooth with a particular structure (*e.g.*, ℓ_1-norm).
- **Quasi-Newton**: how to exploit function curvature with a reasonable cost.

Other ideas and recent contributions

Other important concepts

- **duality gaps**: convex optimization offers mechanisms to control $f(x_t) - f^\star$ in practice!
- **composite optimization**: all previous approaches can be extended to solve

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

  where Ω is convex and non-smooth with a particular structure (*e.g.*, ℓ_1-norm).
- **Quasi-Newton**: how to exploit function curvature with a reasonable cost.

Example of our recent contributions (with H. Lin, Z. Harchaoui, and A. Kulunchakov)

- **Catalyst**: provides Nesterov's acceleration to algorithms [Lin et al., 2018].
- **QNing**: provides Quasi-Newton acceleration to algorithms [Lin et al., 2019].
- **acceleration + variance-reduction + robustness** [Kulunchakov and Mairal, 2019a].

Part II: Generic Acceleration

- H. Lin, J. Mairal, and Z. Harchaoui. Catalyst Acceleration for First-order Convex Optimization: from Theory to Practice. Journal of Machine Learning Research (JMLR). 2018.
- H. Lin, J. Mairal, and Z. Harchaoui. An Inexact Variable Metric Proximal Point Algorithm for Generic Quasi-Newton Acceleration. SIAM Journal on Optimization. 2019.
- A. Kulunchakov and J. Mairal. A Generic Acceleration Framework for Stochastic Composite Optimization. Adv. Neural Information Processing Systems (NeurIPS). 2019.

The Catalyst approach [Lin, Mairal, and Harchaoui, 2018]

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 $\mathcal{M}$ for approximately solving:

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

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

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

see also [Frostig et al., 2015, Devolder et al., 2014, Shalev-Shwartz and Zhang, 2014]

An old idea, apparently unrelated to acceleration

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] ...

An old idea, apparently unrelated to acceleration

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 $f : \mathbb{R}^d \to \mathbb{R}$ 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, ∇F is Lipschitz continuous with parameter $L_F = \kappa$.

- If f is μ-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, ∇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

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

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 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 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 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].

Remarks

- F may be **better conditioned** than f when $1 + \kappa/\mu \leq L/\mu$;
- Computing $p(y_k)$ has a cost!

The Catalyst approach [Lin, Mairal, and Harchaoui, 2018]

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 $\mathcal{M}$ for approximately solving:

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

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

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

see also [Frostig et al., 2015, Devolder et al., 2014, Shalev-Shwartz and Zhang, 2014]

The QNing approach [Lin, Mairal, and Harchaoui, 2019]

Replace Nesterov's acceleration in Catalyst by a Quasi-Newton method (L-BFGS)

$$x_{k+1} = x_k - \eta_k B_k^{-1} g_k \quad \text{with} \quad g_k \approx \nabla F(x_k).$$

The quantity g_k is obtained by using an optimization method $\mathcal{M}$:

$$g_k = \kappa(x_k - z_k) \text{with} \quad z_k \approx \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x_k\|^2 \right\},$$

QNing provides Quasi-Newton principles to $\mathcal{M}$ with...

- **L-BFGS** rules for building B_k.
- **restart strategies** for solving the sub-problems;
- **global complexity analysis** resulting in ... no acceleration;
- **balancing between outer and inner computations** resulting in **practical** acceleration.

see also [Friedlander and Schmidt, 2012, Nocedal, 1980]

Part III: the Cyanure software package

http://julien.mairal.org/cyanure/welcome.html

The Cyanure software package

Binary classification with ℓ_2-logistic regression on the Criteo dataset (21Gb, huge sparse matrix). We use a three-years-old quad-core workstation with 32Gb of memory.

```
import cyanure as cyan
import scipy.sparse
import numpy as np
#load criteo dataset 21Gb, n=45840617, p=999999
dataY=np.load('criteo_y.npz',allow_pickle=True); y=dataY['y']
X = scipy.sparse.load_npz('criteo_X.npz')
#normalize the rows of X in-place, without performing any copy
cyan.preprocess(X,normalize=True,columns=False)
#declare a binary classifier for l2-logistic regression
classifier=cyan.BinaryClassifier(loss='logistic',penalty='l2')
# uses the auto solver by default, performs at most 500 epochs
classifier.fit(X,y,lambd=0.1/X.shape[0],max_epochs=500,tol=1e-3,it0=5)
```

The Cyanure software package

```
Matrix X, n=45840617, p=999999
*********************************
Catalyst Accelerator, MISO Solver, Incremental Solver with uniform sampling
Logistic Loss is used with L2 regularization
Epoch: 5, primal objective: 0.456014, time: 92.5784
Best relative duality gap: 14383.9
Epoch: 10, primal objective: 0.450885, time: 227.593
Best relative duality gap: 1004.69
Epoch: 15, primal objective: 0.450728, time: 367.939
Best relative duality gap: 6.50049
Epoch: 20, primal objective: 0.450724, time: 502.954
Best relative duality gap: 0.068658
Epoch: 25, primal objective: 0.450724, time: 643.323
Best relative duality gap: 0.00173208
Epoch: 30, primal objective: 0.450724, time: 778.363
Best relative duality gap: 0.00173207
Epoch: 35, primal objective: 0.450724, time: 909.426
Best relative duality gap: 9.36947e-05
Time elapsed : 928.114
```

The Cyanure software package

We now learn an SVM with ℓ_1-regularization on this laptop.

```
import cyanure as cyan
import numpy as np
import scipy.sparse
#load rcv1 dataset about 1Gb, n=781265, p=47152
data = np.load('rcv1.npz',allow_pickle=True); y=data['y']; X=data['X']
X = scipy.sparse.csc_matrix(X.all()).T # n x p matrix, csr format
#normalize the rows of X in-place, without performing any copy
cyan.preprocess(X,normalize=True,columns=False)
#declare a binary classifier for squared hinge loss + l1 regularization
classifier=cyan.BinaryClassifier(loss='sqhinge',penalty='l2')
# uses the auto solver by default, performs at most 500 epochs
classifier.fit(X,y,lambd=0.000005,max_epochs=500,tol=1e-3)
```

The Cyanure software package

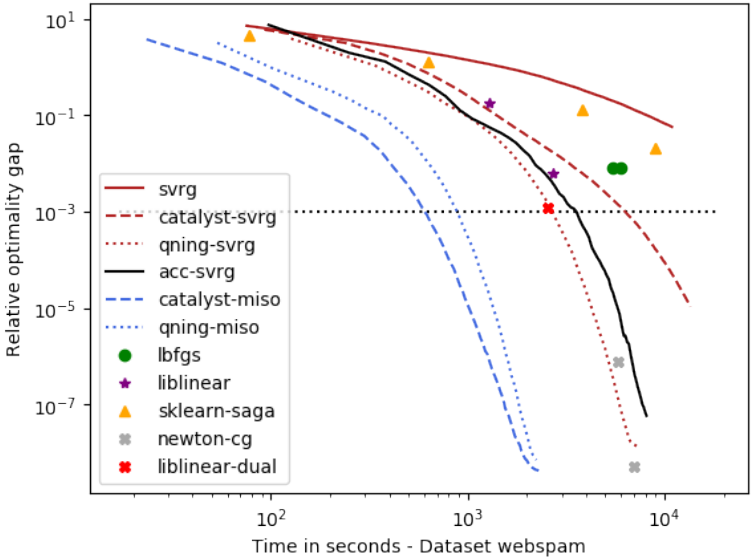
```
Matrix X, n=781265, p=47152
Memory parameter: 20
********************************
QNing Accelerator, MISO Solver
Squared Hinge Loss with L1 regularization
Epoch: 10, primal objective: 0.0915524, time: 7.33038
Best relative duality gap: 0.387338
Epoch: 20, primal objective: 0.0915441, time: 15.524
Best relative duality gap: 0.00426003
Epoch: 30, primal objective: 0.0915441, time: 25.738
Best relative duality gap: 0.000312145
Time elapsed : 26.0225
Total additional line search steps: 8
Total skipping l-bfgs steps: 0
```

Other examples are available on the website.

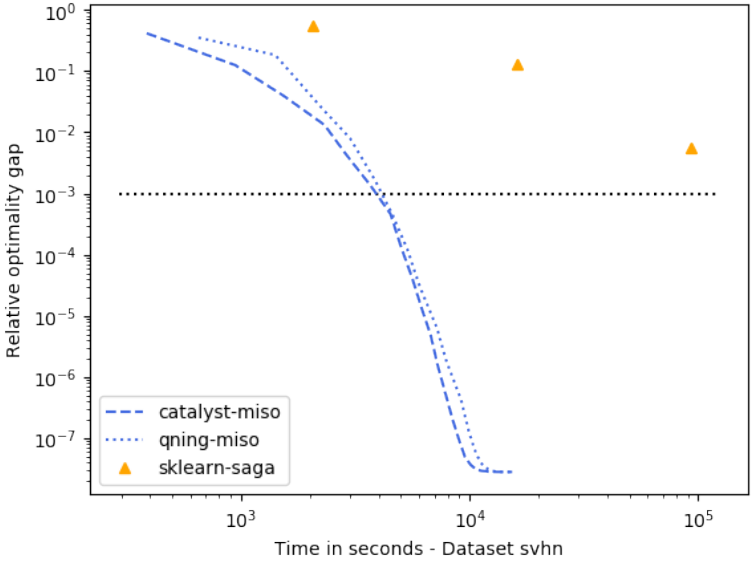
The Cyanure software package, benchmarks

Dataset	Sparse	Num classes	n	p	Size (in Gb)
covtype	No	1	581012	54	0.25
alpha	No	1	500000	500	2
real-sim	No	1	72309	20958	0.044
epsilon	No	1	250000	2000	4
ocr	No	1	2500000	1155	23.1
rcv1	Yes	1	781265	47152	0.95
webspam	Yes	1	250000	16609143	14.95
kddb	Yes	1	19264097	28875157	6.9
criteo	Yes	1	45840617	999999	21
ckn_mnist	No	10	60000	2304	0.55
ckn_svhn	No	10	604388	18432	89

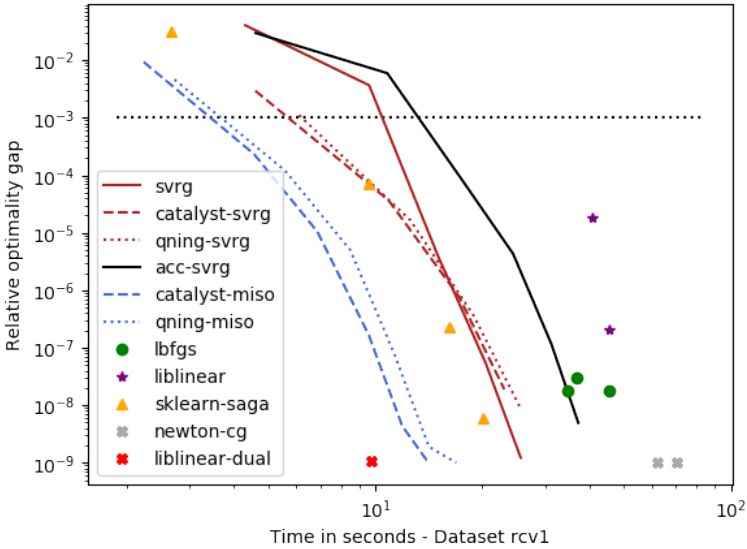
The Cyanure software package, benchmarks



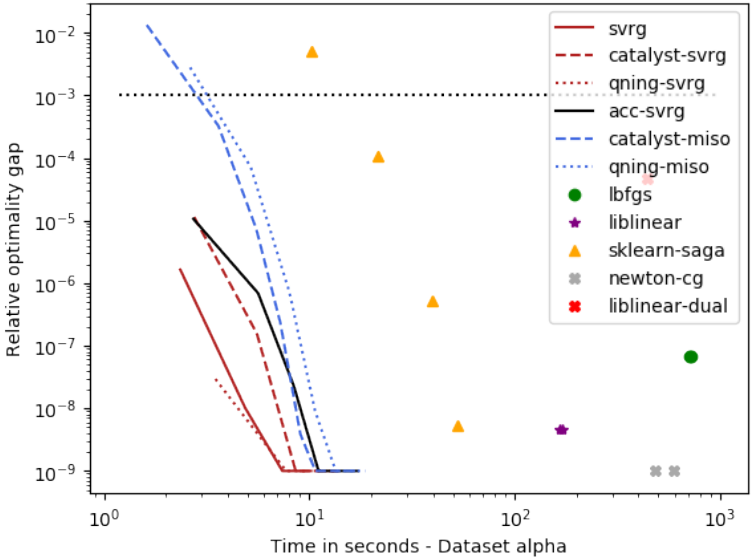
The Cyanure software package, benchmarks



The Cyanure software package, benchmarks



The Cyanure software package, benchmarks



Conclusion

Challenges for algorithms

- going beyond the comfortable convex setting with i.i.d. data.
- better exploit the function curvature for nonconvex problems.
- extension to stochastic optimization [Kulunchakov and Mairal, 2019b].

Conclusion

Challenges for algorithms

- going beyond the comfortable convex setting with i.i.d. data.
- better exploit the function curvature for nonconvex problems.
- extension to stochastic optimization [Kulunchakov and Mairal, 2019b].

Challenges for Cyanure

Cyanure is still in its early stage. Do not hesitate to post issues/request on github.

Todo list

- Interface for R and Matlab.
- Improve scikit-learn compatibility.
- ...

Any suggestion is welcome.

References I

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

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.

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

David Corfield, Bernhard Schölkopf, and Vladimir Vapnik. Falsificationism and statistical learning theory: Comparing the popper and vapnik-chervonenkis dimensions. *Journal for General Philosophy of Science*, 40(1):51–58, 2009.

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)*, 2014.

Olivier Devolder, F. Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

References II

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

Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

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

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.

Andrei Kulunchakov and Julien Mairal. Estimate sequences for stochastic composite optimization: Variance reduction, acceleration, and robustness to noise. *arXiv preprint arXiv:1901.08788*, 2019a.

Andrei Kulunchakov and Julien Mairal. A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

References III

Claude Lemaréchal and Claudia Sagastizábal. Practical aspects of the moreau–yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

H. Lin, J. Mairal, and Z. Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *Journal of Machine Learning Research (JMLR)*, 18(212):1–54, 2018.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. An inexact variable metric proximal point algorithm for generic quasi-newton acceleration. *SIAM Journal on Optimization*, 29(2):1408–1443, 2019.

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

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Revue franÃ§aise d'informatique et de recherche opérationnelle, série rouge*, 1970.

Robert Mifflin. A quasi-second-order proximal bundle algorithm. *Mathematical Programming*, 73(1): 51–72, 1996.

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.

References IV

Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

R. T. 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.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pages 1–41, 2014.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1995.

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