Master 2 Data Science, Univ. Paris Saclay

Optimization for Data Science

Stéphane Gaïffas

We want to minimize

$$F(x) = f(x) + g(x)$$

where f is goodness-of-fit

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \text{with} \quad f_i(x) = \ell(b_i, \langle a_i, x \rangle)$$

and g is penalization, where main examples are

$$g(x) = \frac{\lambda}{2}\|x\|_2^2 \quad \text{(ridge)} \qquad g(x) = \lambda\|x\|_1 \quad \text{(lasso)}$$

At each iteration **gradient descent (GD)** methods use

$$\nabla f(x) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x),$$

while **stochastic gradient descent (SGD)** use

$$\nabla f_I(x)$$

where $I \in \{1, \ldots, n\}$ is chosen uniformly at random

Remark. $\nabla f_I(x)$ is an unbiased but very noisy estimate of the full gradient $\nabla f(x)$

Stochastic Gradient Descent

Input: starting point x_0, steps (learning rates) η_t

For $t = 1, 2, \ldots$ until *convergence* do

- Pick at random (uniformly) i_t in $\{1, \ldots, n\}$
- compute

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

Return last x_t

Rate of convergence of GD versus SGD is

$$O\Big(\frac{n}{\mu} \log\big(\frac{1}{\varepsilon}\big)\Big) \quad \text{versus} \quad O\Big(\frac{1}{\mu\varepsilon}\Big)$$

if f is μ-strongly convex

Remarks.

- SGD is very fast in the early iterations
- SGD does n iterations while GD does nothing
- If ε small $1/\varepsilon \gg \log(1/\varepsilon)$ hard for SGD to converge to a precise solution

Why?

- Stochastic gradients are unbiased but have a large variance

How to improve this?

- Use a variance reduction technique

Recent results improve this:

- Bottou and LeCun (2005)
- Shalev-Shwartz et al (2007, 2009)
- Nesterov et al. (2008, 2009)
- Bach et al. (2011, 2012, 2014, 2015)
- T. Zhang et al. (2014, 2015)

The problem.

- Put $X = \nabla f_I(x)$ with I uniformly chosen at random in $\{1, \ldots, n\}$
- In SGD we use $X = \nabla f_I(x)$ as an approximation of $\mathbb{E}X = \nabla f(x)$
- How to reduce var X ?

An idea

- Reduce it by finding C s.t. $\mathbb{E}C$ is "easy" to compute and such that C is highly correlated with X
- Put $Z_\alpha = \alpha(X - C) + \mathbb{E}C$ for $\alpha \in [0, 1]$. We have

$$\mathbb{E}Z_\alpha = \alpha\mathbb{E}X + (1 - \alpha)\mathbb{E}C$$

and

$$\operatorname{var} Z_\alpha = \alpha^2(\operatorname{var} X + \operatorname{var} C - 2\operatorname{cov}(X, C))$$

- Standard variance reduction: $\alpha = 1$, so that $\mathbb{E}Z_\alpha = \mathbb{E}X$ (unbiased)

Variance reduction of the gradient

In the iterations of SGD, replace $\nabla f_{i_t}(x_{t-1})$ by

$$\alpha(\nabla f_{i_t}(x_{t-1}) - \nabla f_{i_t}(\widetilde{x})) + \nabla f(\widetilde{x})$$

where $\widetilde{x}$ is an "old" value of the iterate, namely use

$$x_t \leftarrow x_{t-1} - \eta\big(\alpha(\nabla f_{i_t}(x_{t-1}) - \nabla f_{i_t}(\widetilde{x})) + \nabla f(\widetilde{x})\big)$$

Several cases

- $\alpha = 1/n$: SAG (Bach et al. 2013)
- $\alpha = 1$: SVRG (T. Zhang et al. 2015, 2015)
- $\alpha = 1$: SAGA (Bach et al., 2014)

Stochastic Average Gradient

Input: starting point x_0, learning rate $\eta > 0$
For $t = 1, 2, \ldots$ until *convergence* do

- Pick uniformly at random i_t in $\{1, \ldots, n\}$
- Put

$$g_t(i) = \begin{cases} \nabla f_i(x_{t-1}) & \text{if } i = i_t \\ g_{t-1}(i) & \text{otherwise} \end{cases}$$

and compute

$$x_t = x_{t-1} - \frac{\eta}{n} \sum_{i=1}^{n} g_t(i)$$

Return last x_t

Stochastic Variance Reduced Gradient

Input: starting point x_0, learning rate $\eta > 0$
Put $\widetilde{x}^1 \leftarrow x_0$
For $k = 1, 2, \ldots$ until *convergence* do

- Put $x_0^k \leftarrow \widetilde{x}^1$
- Compute $\nabla f(\widetilde{x}^k)$
- For $t = 0, \ldots, m-1$
  - Pick uniformly at random i in $\{1, \ldots, n\}$
  - Apply the step

$$x_{t+1}^k \leftarrow x_t^k - \eta(\nabla f_i(x_t^k) - \nabla f_i(\widetilde{x}^k) + \nabla f(\widetilde{x}^k))$$

- Set

$$\widetilde{x}^k \leftarrow \frac{1}{m}\sum_{t=1}^{m} x_t^k$$

Return last x_t^k

SAGA

Input: starting point x_0, learning rate $\eta > 0$
Compute $g_0(i) \leftarrow \nabla f_i(x_0)$ for all $i = 1, \ldots, n$
For $t = 1, 2, \ldots$ until *convergence* do

- Pick uniformly at random i_t in $\{1, \ldots, n\}$
- Compute $\nabla f_{i_t}(x_{t-1})$
- Apply

$$x_t \leftarrow x_{t-1} - \eta\left(\nabla f_{i_t}(x_{t-1}) - g_{t-1}(i_t) + \frac{1}{n}\sum_{i=1}^{n} g_{t-1}(i)\right)$$

- Store $g_t(i_t) \leftarrow \nabla f_{i_t}(x_{t-1})$

Return last x_t

Stochastic Variance Reduced Gradient

Phase size typically chosen as $m = n$ or $m = 2n$

If $F = f + g$ with g prox-capable, use

$$x_{t+1}^k \leftarrow \text{prox}_{\eta g}(x_t^k - \eta(\nabla f_i(x_t^k) - \nabla f_i(\widetilde{x}^k) + \nabla f(\widetilde{x}^k)))$$

SAGA

If $F = f + g$ with g prox-capable, use

$$x_t \leftarrow \text{prox}_{\eta g}\left(x_{t-1} - \eta\left(\nabla f_{i_t}(x_{t-1}) - g_{t-1}(i_t) + \frac{1}{n}\sum_{i=1}^{n} g_{t-1}(i)\right)\right)$$

Important remark

- In these algorithms, the step-size η is kept **constant**
- Leads to **linearly convergent algorithms**, with a numerical complexity comparable to SGD!

Theoretical guarantees

- Each f_i is L_i-smooth. Put $L_{\max} = \max_{i=1,\dots n} L_i$
- f is μ-strongly convex

For SAG

Take $\eta = 1/(16L_{\max})$ constant

$$\mathbb{E}f(x_t) - f(x_*) \leq O\Big(\frac{1}{n\mu} + \frac{L_{\max}}{n}\Big) \exp\Big(-t\Big(\frac{1}{8n} \wedge \frac{\mu}{16L_{\max}}\Big)\Big)$$

The rate is typically faster than gradient descent!

For SVRG

Take η and m such that

$$\rho = \frac{1}{1 - 2\eta L_{\max}} \left(\frac{1}{m\eta\mu} + 2L_{\max}\eta \right) < 1$$

Then

$$\mathbb{E} f(x^k) - f(x_*) \leq \rho^k (f(x^0) - f(x_*))$$

[we will prove that later...]

In practice $m = n$ and $\eta = 1/L_{\max}$ works

In summary, about variance reduction

- Complexity $O(d)$ instead of $O(nd)$ at each iteration
- Choice of a **fixed** step-size $\eta > 0$ possible
- Much faster than full gradient descent!

Numerical complexities

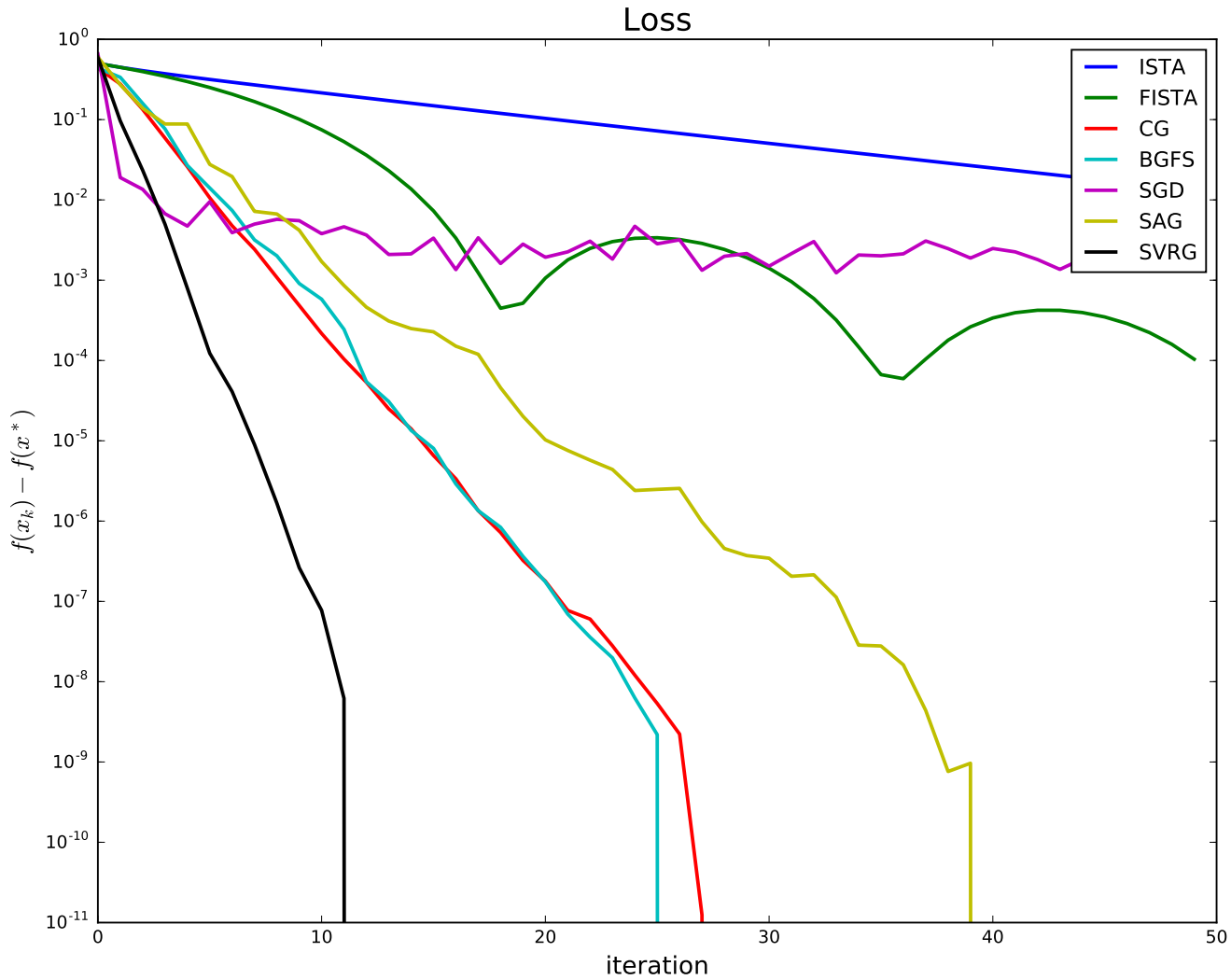
- $O(nL/\mu \log(1/\varepsilon))$ for GD
- $O(1/(\mu n))$ for SGD
- $O((n + L_{\max}/\mu) \log(1/\varepsilon))$ for SGD with variance reduction (SAG, SAGA, SVRG, etc.)

where $L =$ Lipschitz constant of $\frac{1}{n}\sum_{i=1}^{n} f_i$.
Note that typically

$$n\frac{L}{\mu}\log(1/\varepsilon) \gg \left(n + \frac{L_{\max}}{\mu}\right)\log(1/\varepsilon)$$

Stochastic VS deterministic solvers

(This is what you will do next week)



Some practical aspects

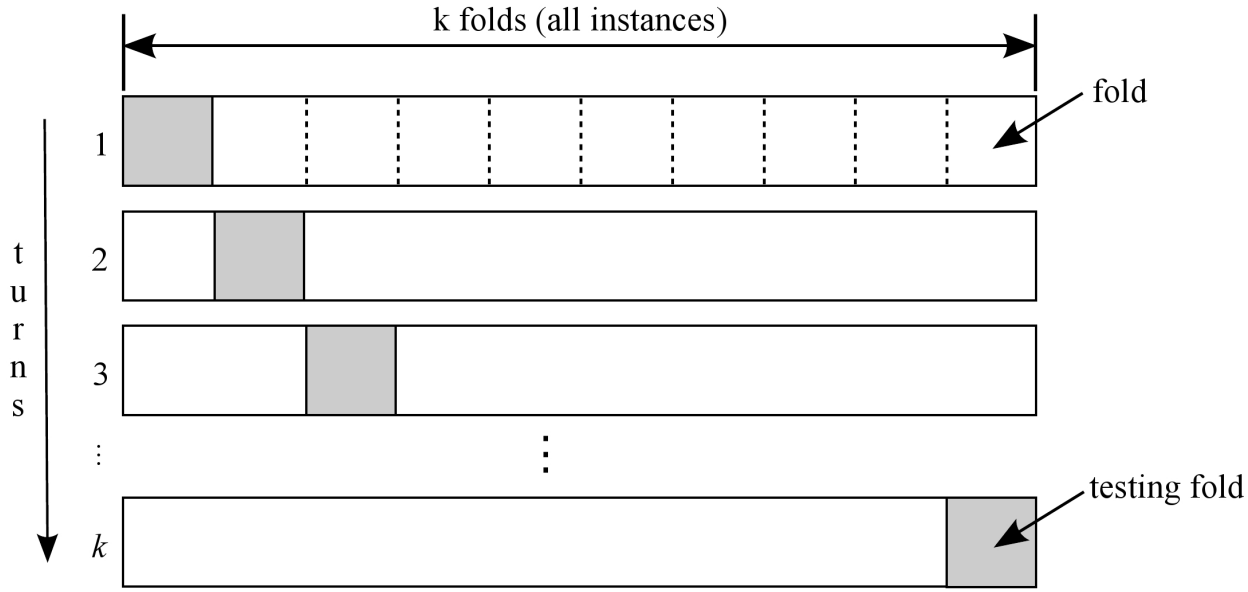
- SAG and SAGA requires extra memory: need to save all the previous gradients!
- Actually no...

$$\nabla f_i(x) = \ell'(b_i, \langle a_i, x \rangle) a_i,$$

  so only need to save $\ell'(b_i, \langle a_i, x \rangle)$
- Memory footprint is $O(n)$ instead of $O(nd)$. If $n = 10^7$, this is 76 Mo
- Same lazy updating trick as for SGD (last lecture)

Some practical aspects

- V-fold cross-validation
- Take $V = 5$ or $V = 10$. Pick a random partition $I_1, \ldots, I_V$ of $\{1, \ldots, n\}$, where $|I_v| \approx \frac{n}{V}$ for any $v = 1, \ldots, V$



How to do it with SGD type algorithms?

Some practical aspects

- V-fold cross-validation

Simple solution

When picking a line i at random in the optimization loop, its fold number is given by $i\%V$

- Pick i uniformly at random in $\{1, \ldots, n\}$
- Put $v = i\%V$
- For $v' = 1, \ldots, V$ with $v' \neq v$: update $\hat{x}^{(v')}$ using line i
- Update the testing error of $\hat{x}^{(v)}$ using line i

Some practical aspects

We want to minimize a sequence of objectives

$$f(x) + \lambda g(x)$$

for $\lambda = \lambda_1, \ldots, \lambda_M$, and select the best using V-fold cross-validation

Idea

Use the fact that solutions $\hat{x}^{\lambda_{j-1}}$ and $\hat{x}^{\lambda_j}$ are close when λ_{j-1} and λ_j are

Warm-starting

Put $x_0 = 0$ (I don't know where to start)

For $m = M, \ldots, 1$

- Put $\lambda = \lambda_m$
- Solve the problems starting at x_0 for this value of λ (on each fold)
- Keep the solutions $\hat{x}$ (test it, save it...)
- Put $x_0 \leftarrow \hat{x}$

This allows to solve much more rapidly the sequence of problems

[Convergence proof for SVRG on the blackboard]