Continuous (convex) optimisation

M2 - PSL / Dauphine / S.U.

Antonin Chambolle, CNRS, CEREMADE

Université Paris Dauphine PSL

Sep.-Nov. 2024

Lecture 7: Large scale problems, stochastic methods.

Contents

Alternating minimization?

Problem:

$$\min_{x_1,\dots,x_n} f(x_1,\dots,x_n)$$

Assume we know how to solve, for $i = 1,\dots,n$ and given $(x_j)_{j\neq i}$:

$$\min_{\xi} f(x_1,\dots,x_{i-1},\xi,x_{i+1},\dots,x_n).$$

Then, the following algorithm is natural:
Let (x^0) be given and for $k \geq 0$, $i = 1,\dots,n$ let:

$$x_i^{k+1} \in \arg\min_{\xi} f(x_1^{k+1},\dots,x_{i-1}^{k+1},\xi,x_{i+1}^{k},\dots,x_n^{k}). \tag{1}$$

Convergence?

Counterexample

For $x = (x_1, x_2) \in \mathbb{R}^2$ let $f(x_1, x_2) = x_1^2/2 + |x_1 - x_2|$.
Then $f(0,0) = 0$ is minimal.

From (x_1^k, x_2^k), the algorithm will first produce $x_1^{k+1} = \max\{-1, \min\{x_2^k, 1\}\}$ and then $x_2^{k+1} = x_1^{k+1}$.

Hence, one has $x_1^k = x_2^k = x_2^1$ for any $k \geq 1$ and unless $x_2^0 = 0$, this never converges to the minimizer.

Alternating minimization?

Assume on the other hand that:

- The space is finite-dimensional;
- f is C^1, bounded from below, coercive ($f(x) \to +\infty$ if $|x| \to \infty$);
- f is convex.

First, one has that $f(x^{k+1}) \leq f(x^k)$ so in particular there is a value f^* with $f(x^k) \to f^* = \inf_k f(x^k)$.

Then, (x^k) is bounded and has a subsequence (x^{k_l}) which converges to some x. Up to a further subsequence, $x^{k_l+1} \to y$. One can easily show that:

$$
\begin{aligned}
f^* = f(y_1, \dots, y_{i-1}, y_i, x_{i+1}, \dots, x_n) &= \min_{\xi} f(y_1, \dots, y_{i-1}, \xi, x_{i+1}, \dots, x_n) \\
&\leq f(y_1, \dots, y_{i-1}, x_i, x_{i+1}, \dots, x_n) = f^*
\end{aligned}
$$

for all i. In particular $\partial_i f(y_1, \dots, y_{i-1}, y_i, x_{i+1}, \dots, x_n) = \partial_i f(y_1, \dots, y_{i-1}, x_i, x_{i+1}, \dots, x_n) = 0$.

Alternating minimization

If $x^k \to x$, one deduces that x is a minimizer, otherwise it is not even clear. Yet using that f is convex, we can show that limit points are minimizers.

Proof: This is shown by induction: let us assume that $\partial_j f(y_1, \dots, y_i, x_{i+1}, \dots, x_n) = 0$ for $j = 1, \dots, i$, $i \leq n - 1$. This is true for $i = 1$.

Now, we have by minimality that $\partial_{i+1} f(y_1, \dots, y_{i+1}, x_{i+2}, \dots) = 0$, and since it has the same value, also $\partial_{i+1} f(y_1, \dots, y_i, x_{i+1}, x_{i+2}, \dots) = 0$.

As a consequence, thanks to the induction hypothesis, $(y_1, \dots, y_i, x_{i+1})$ is a minimizer of the convex function $f(\bullet, x_{i+2}, \dots, x_n)$ and since it has the same value, also $(y_1, \dots, y_{i+1})$ is a minimizer. It follows that $\partial_j f(y_1, \dots, y_{i+1}, x_{i+2}, \dots, x_n) = 0$ for all $j \leq i + 1$, which shows the induction.

As a consequence, $\partial_j f(y) = 0$ for all j and y is a minimizer of f. Since x has the same value and it is also a minimizer of f. □

(Block) coordinate descent

We can replace the minimization with a step of gradient descent. If f has Lipschitz gradients:

$$x_i^{k+1} = x_i^k - \tau_i \nabla_i f(x_1^{k+1}, \dots, x_{i-1}^{k+1}, x_i^k, \dots, x_n^k).$$

Here, $\nabla_i := \partial / \partial x_i$.

(Block) coordinate descent

Assume that $\partial_i f$ is L_i-Lipschitz (uniformly): as usual,

$$f(x_1^{k+1},\dots,x_i^{k+1},x_{i+1}^k,\dots,x_n^k) \le f(x_1^{k+1},\dots,x_{i-1}^{k+1},x_i^k,\dots,x_n^k) \\ - \tau_i(1-\tfrac{L_i\tau_i}{2})|\nabla_i f(x_1^{k+1},\dots,x_{i-1}^{k+1},x_i^k,\dots,x_n^k)|^2$$

Choosing $\tau_i = \frac{1}{L_i}$:

$$f(x_1^{k+1},\dots,x_i^{k+1},x_{i+1}^k,\dots,x_n^k) + \frac{1}{2L_i}|\nabla_i f(x_1^{k+1},\dots,x_{i-1}^{k+1},x_i^k,\dots,x_n^k)|^2 \\ \le f(x_1^{k+1},\dots,x_{i-1}^{k+1},x_i^k,\dots,x_n^k)$$

$\rightarrow$ as in the previous analysis, in the convex case one deduces that limit points are minimizers.

(Block) Coordinate descent

One interesting point here is that in general, the Lipschitz constant with respect to one variable is smaller than with respect to all the variables

Example: $(x_1, x_2) \mapsto (x_1 + x_2)^2$ has $\sqrt{2}$-Lipschitz gradient but the partial gradients are 1-Lipschitz.

$\rightarrow$ longer steps.

Variants: change the order of updates. Random order.

Random coordinate descent

Algorithm: choose x^0.
At iteration $k \geq 0$, choose $i_k \in \{1, \ldots, n\}$ randomly with probablilities $(p_1, \ldots, p_n)$ $(\sum_i p_i = 1)$. Then let:

$$\begin{cases} x_{i_k}^{k+1} = x_{i_k}^k - \tau_{i_k} \nabla_{i_k} f(x^k), \\ x_j^{k+1} = x_j^k & \text{for } j \neq i_k. \end{cases}$$

Random coordinate descent

Algorithm: choose x^0.
At iteration $k \geq 0$, choose $i_k \in \{1, \ldots, n\}$ randomly with probablilities $(p_1, \ldots, p_n)$ $(\sum_i p_i = 1)$. Then let:

$$\begin{cases} x_{i_k}^{k+1} = x_{i_k}^k - \tau_{i_k} \nabla_{i_k} f(x^k), & \\ x_j^{k+1} = x_j^k & \text{for } j \neq i_k. \end{cases}$$

We have, given x^k and i_k:

$$f(x^{k+1}) \leq f(x^k) - \tau_{i_k}(1 - \tfrac{L_{i_k}\tau_{i_k}}{2})|\nabla_{i_k} f(x^k)|^2 \tag{2}$$

As a consequence, knowing the point x^k, the expectation $\mathbb{E}(f(x^{k+1})|x^k)$ satisfies

$$\mathbb{E}(f(x^{k+1})|x^k) \leq f(x^k) - \sum_{i=1}^n p_i \tau_i (1 - \tfrac{L_i \tau_i}{2})|\nabla_i f(x^k)|^2.$$

Random coordinate descent

Let $\tau_i = 1/L_i$ and $p_i = L_i/(\sum_j L_j)$ (we pick more often the coordinates with larger Lipschitz constants). Then:

$$\mathbb{E}(f(x^{k+1})|x^k) \le f(x^k) - \frac{1}{2\sum_j L_j}\sum_{i=1}^n |\nabla_i f(x^k)|^2 = f(x^k) - \frac{1}{2\sum_j L_j}|\nabla f(x^k)|^2.$$

Then we compute the expectation with respect to x^k:

$$\mathbb{E}(f(x^{k+1})) \le \mathbb{E}(f(x^k)) - \frac{1}{2\sum_j L_j}\mathbb{E}(|\nabla f(x^k)|^2). \quad (3)$$

In particular, $\mathbb{E}(f(x^k))$ is a decreasing sequence, and one has

$$\frac{1}{2\sum_j L_j}\sum_{k=0}^{\infty}\mathbb{E}(|\nabla f(x^k)|^2) \le f(x^0) < \infty$$

which shows that $\mathbb{E}(|\nabla f(x^k)|^2) \to 0$ (up to a subsequence $\nabla f(x^k) \to 0$ a.s.).

Random coordinate descent

More generally: pick $\tau_i = \theta/L_i$ for $\theta \in]0,2[$, let $|g|_M^2 := \sum_{i=1}^n m_i|g_i|^2$, for $m_i := p_i/L_i$. Then with the same computation we get:

$$\mathbb{E}(f(x^{k+1})|x^k) \le f(x^k) - \sum_{i=1}^n \frac{\theta(2-\theta)p_i}{L_i}|\nabla_i f(x^k)|^2 = f(x^k) - \frac{\theta(2-\theta)}{2}|\nabla f(x^k)|_M^2.$$

If we assume that there exists a minimizer x^*, let $\Delta_k := f(x^k) - f(x^*)$. Then:

Lemma

Assume $\{f \le f(x^0)\}$ *is bounded. Then*

$$\mathbb{E}(\Delta_k) \le \frac{2D^2}{\theta(2-\theta)}\frac{1}{k+1}$$

where $D \ge \sup_{f(x)\le f(x^0)} |x - x^*|_{M^{-1}}$.[1]

[1] The traditional "L" constant is here included in the norm $|\cdot|_M$.

Random coordinate descent

Proof. As usual from the convexity of f we get:

$$f(x) - f(x^*) \le \langle \nabla f(x), x - x^* \rangle \le |\nabla f(x)|_M |x^* - x|_{M^{-1}} \le D |\nabla f(x)|_M$$

if $f(x) \le f(x^0)$ and D is as in the Lemma. Then:

$$\mathbb{E}(f(x^{k+1}) - f(x^*) | x^k) \le f(x^k) - f(x^*) - \frac{\theta(2-\theta)}{2} \frac{(f(x^k) - f(x^*))^2}{D^2}.$$

By convexity (using Jensen's inequality): $\mathbb{E}(\Delta_k)^2 \le \mathbb{E}(\Delta_k^2)$, hence:

$$\mathbb{E}(\Delta_{k+1}) \le \mathbb{E}(\Delta_k) - \frac{\theta(2-\theta)}{2D^2} \mathbb{E}(\Delta_k^2) \le \mathbb{E}(\Delta_k) - \frac{\theta(2-\theta)}{2D^2} \mathbb{E}(\Delta_k)^2.$$

Then we conclude as for the standard gradient descent. □

Random coordinate descent

Comments:

- Ideally the probabilities should minimize the “diameter” D...
- Standard choice already mentioned: $\theta = 1$, $p_i = L_i / \sum_j L_j$. Then the rate becomes:

$$\mathbb{E}(\Delta_{nk}) \leq \left(\frac{2}{n} \sum_{j=1}^{n} L_j \right) \frac{\sup_{f(x) \leq f(x^0)} |x - x^*|^2}{k + 1/n}$$

after k “epochs” (that is nk iterations, or k average passes over all the variables).

$$\mathbb{E}(\Delta_{nk}) \leq \left(\frac{2}{n}\sum_{j=1}^{n} L_j\right) \frac{\sup_{f(x)\leq f(x^0)} |x - x^*|^2}{k + 1/n}$$

This is to be compared to the rate for deterministic G.D.:

$$\Delta_k \leq 2L\frac{|x^0 - x^*|^2}{k+1}$$

now L is the global Lipschitz constant of f: we have replaced L with $\bar{L} := (1/n)\sum_j L_j$.

Random coordinate descent

Comparison with gradient descent

One always has:

$$\max_j L_j \leq L \leq \sqrt{\sum_{j=1}^n L_j^2},$$

and in particular $\bar{L} \leq L$. On the other hand:

$$\bar{L} = \frac{1}{n}\sum_j L_j \leq \frac{1}{\sqrt{n}}\sqrt{\sum_{j=1}^n L_j^2}.$$

One always has:

$$\max_j L_j \leq L \leq \sqrt{\sum_{j=1}^n L_j^2},$$

and in particular $\bar{L} \leq L$. On the other hand:

$$\bar{L} = \frac{1}{n}\sum_j L_j \leq \frac{1}{\sqrt{n}}\sqrt{\sum_{j=1}^n L_j^2}.$$

- In the worst case, the complexity of the random coordinate descent is similar to the deterministic gradient descent;
- If L is close to the upper bound $\sqrt{\sum_j L_j^2}$ then the complexity might be smaller by a factor up to $1/\sqrt{n}$ (where n is the number of coordinates).

Random coordinate descent

Extensions, variants

- For minimizing: $f(x) + \sum_{i=1}^{n} \psi_i(x_i)$, one can replace the kth iteration with the proximal iteration

$$x_{i_k}^{k+1} = (I + \tau_{i_k} \partial \psi_i)^{-1}(x_{i_k} - \tau_{i_k} \nabla_{i_k} f(x^k))$$

with $\tau_i = 1/L_i$. Then one gets similar results (Richtárik, Takáč, Math. Program. 144, 2014).

- Acceleration: Fercoq, Richtárik, "Approx" algorithm (SIAM Rev. 58, 2016).

Stochastic gradient descent

Typical "learning" problem (such as "SVM", see later): minimize (for large $n \geq 1$) a sum of convex functions:

$$\min_x \frac{1}{n} \sum_i f_i(x) + \psi(x)$$

If ψ is strongly convex, one can derive a dual problem

$$\max_{y_1,\dots,y_n} -\frac{1}{n} \sum_i f_i^*(y_i) - \psi^*(-\tfrac{1}{n} \sum_i y_i)$$

with now ψ^* with Lipschitz gradient: proximal variant random coordinate descent algorithm (previous slide). (See also "stochastic dual coordinate ascent" methods, Shalev-Shwartz and Zhang 2013 [SDCA], 2016 [PSDCA] with acceleration.)

Stochastic gradient descent

For a smooth $f = \frac{1}{n}\sum_i f_i$ (and without ψ), one can use the gradient descent but if n is too large it might not be a good idea to evaluate ∇f at each iteration.

Algorithm ("SGD"): choose x^0. For each $k \geq 1$, choose $\tau > 0$ and pick $i_k \in \{1, \dots, n\}$ with probability $1/n$. Let:

$$x^{k+1} = x^k - \tau \nabla f_{i_k}(x^k).$$

The general idea is that $x^{k+1} = x^k - \tau g_k$ where g_k is a random process with $\mathbb{E}(g_k|x^k) = \nabla f(x^k)$, hence the term "stochastic gradient". Indeed for the choice $g_k(x^k) = \nabla f_{i_k}(x^k)$ with probability $1/n$, one has $\mathbb{E}(g_k|x^k) = \sum_i \frac{1}{n}\nabla f_i(x^k) = \nabla f(x^k)$.

Stochastic gradient descent

One has: $\mathbb{E}(x^{k+1}|x^k) = x^k - \tau\mathbb{E}(g_k|x_k) = x^k - \tau\nabla f(x^k)$.

As usual, one can write that for $j = 1, \dots, n$, if $i_k = i$,

$$f_j(x^{k+1}) \leq f_j(x^k) - \tau \left\langle \nabla f_j(x^k), g_k \right\rangle + \frac{L_j\tau^2}{2}|g_k|^2$$

and summing (and $/n$):

$$f(x^{k+1}) \leq f(x^k) - \tau \left\langle \nabla f(x^k), g^k \right\rangle + \frac{\tau^2}{2}\left(\frac{1}{n}\sum_{j=1}^{n} L_j\right)|g_k|^2.$$

Stochastic gradient descent

Now we can compute the expectation knowing x^k, using that

$$\mathbb{E}(g_k|x^k) = \nabla f(x^k),$$
$$\mathbb{E}(|g_k|^2\,|x^k) = \mathbb{E}(|g_k - \nabla f(x^k)|^2\,|x^k) + |\nabla f(x^k)|^2 = \mathrm{Var}(g_k|x^k) + |\nabla f(x^k)|^2.$$

We find, with $\bar{L} := (1/n)\sum_j L_j$:

$$\mathbb{E}(f(x^{k+1})|x^k) \le f(x^k) - \tau(1 - \tfrac{\tau\bar{L}}{2})|\nabla f(x^k)|^2 + \frac{\tau^2\bar{L}}{2}\mathrm{Var}(g_k|x^k).$$

Problem: for $\tau < 2/\bar{L}$, one expects that $\mathbb{E}(f(x^k))$ decreases until $\mathbb{E}(|\nabla f(x^k)|^2)$ (which is of the order of $|x^k - x^{k+1}|^2$) becomes comparable to $\tau\times$ the variance. Hence one needs either:

- to decrease τ at each step (Robbins, Monro, 1951);
- to find tricks to "reduce" the variance (SAG, SAGA: Le Roux, Schmidt, Bach 2012, Defazio, Bach, Lacoste-Julien 2014, SVRG: Xiao, Zhang, 2014).

Stochastic gradient descent

Robbins, Monro 1951: reduce the step size. If we assume we have an estimate of

$$\mathrm{Var}(g(x)) \leq \sigma^2$$

for x close to x^* (provided we could show that x^k remains close to x^*! which is not a priori clear...)
Then, for $\tau_k \leq 1/\bar{L}$:

$$\left(\sum_{k=0}^{n-1} \tau_k\right) \min_{k=0,\dots,n-1} \mathbb{E}(|\nabla f(x^k)|^2) \leq f(x^0) + \frac{\bar{L}}{2}\sigma^2 \sum_{k=0}^{n-1} \tau_k^2$$

so that:

$$\min_{k=0,\dots,n-1} \mathbb{E}(|\nabla f(x^k)|^2) \leq \frac{f(x^0) + \frac{\bar{L}}{2}\sigma^2 \sum_{k=0}^{n-1} \tau_k^2}{\sum_{k=0}^{n-1} \tau_k}.$$

Stochastic gradient descent

One obtains a rate which is governed by the ratio:

$$\frac{\sum_{k=0}^{n-1} \tau_k^2}{\sum_{k=0}^{n-1} \tau_k}.$$

For instance: $\tau_k \sim 1/k$, the rate is $\sim C/\log n$, for $\tau_k \sim 1/\sqrt{k}$, the rate is $\sim C \log n/\sqrt{n}$.

This is nearly optimal: if one knew all the parameters of the problem and fixed the number of iterations, then letting $\bar{L}\sigma^2 n\tau^2/2 = f(x^0)$, we get:

$$\min_{k=0,\dots,n-1} \mathbb{E}(|\nabla f(x^k)|^2) \leq \frac{f(x^0) + \frac{\bar{L}}{2}\sigma^2 n\tau^2}{n\tau} = \frac{\sqrt{2\bar{L}f(x^0)}}{\sqrt{n}}\sigma$$

Stochastic gradient descent

Reduced variance method

Simplest approach: mini-batching: one can reduce the variance by computing *several* gradients simultaneously (but of course it is then more expensive, with the full gradient as an extreme case and 0 variance)

Stochastic gradient descent

Reduced variance method

Simplest approach: mini-batching: one can reduce the variance by computing *several* gradients simultaneously (but of course it is then more expensive, with the full gradient as an extreme case and 0 variance)

More refined: SAGA (Defazio, Bach, Lacoste-Julien, NeurIPS 2014): the idea is to replace g_k with an *unbiased* (that is $\mathbb{E}(g_k|x^k) = \nabla f(x^k)$) approximation of the gradient with a smaller variance, of the form:

$$g_k = \nabla f_{i_k}(x_k) - v_{i_k} + \frac{1}{n}\sum_j v_j$$

for some $v \approx \nabla f$ depending on the previous iterates.

One has

$$\mathbb{E}(g_k|x^k) = \nabla f(x^k) - \tfrac{1}{n}\sum_i v_i + \tfrac{1}{n}\sum_j v_j = \nabla f(x^k)$$

and

$$\begin{aligned}\mathrm{Var}(g_k|x^k) &= \tfrac{1}{n}\sum_i \Big|\nabla f_i(x^k) - v_i - \tfrac{1}{n}\sum_j(\nabla f_j(x^k) - v_j)\Big|^2 \\ &= \tfrac{1}{n}\sum_i |\nabla f_i(x^k) - v_i|^2 - \Big|\tfrac{1}{n}(\sum_j(\nabla f_j(x^k) - v_j)\Big|^2 \\ &\le \tfrac{1}{n}\sum_i |\nabla f_i(x^k) - v_i|^2\end{aligned}$$

which gets small if v_i is close to $\nabla f_i(x^k)$. But v_i should not depend on i_k (only on the past) and of course, the “ideal” choice $v_i = \nabla f_i(x^k)$ consists in computing the full gradient at each step.

In the SAGA method v_i is the last computed value of $\nabla f_i(x^l)$, at a previous iterate l.

Algorithm (SAGA): choose x^0, $v_i = 0$, $\bar{v} = 0$.

1. for each $k \geq 0$: pick $i_k \in \{1, \dots, n\}$ with probability $1/n$.
2. Let $v_{\text{old}} = v_{i_k}$;
3. Let $v_{i_k} = \nabla f_{i_k}(x_k)$ ("new");
4. let $x^{k+1} = x^k - \tau(v_{i_k} - v_{\text{old}} + \bar{v})$;
5. let $\bar{v} = \bar{v} + \frac{1}{n}(v_{i_k} - v_{\text{old}})$.

One sees that at each iteration, $\bar{v}$ is kept to $\frac{1}{n}\sum_j v_j$.

Stochastic gradient descent

Reduced variance method: SAGA

Rate for SAGA [Defazio, Bach, Lacoste-Julien, NeurIPS 2014]:

If the f_i's have L-Lipschitz gradient, then for $\tau = 1/(3L)$, one has, letting $\bar{x}^k := (1/k)\sum_{t=1}^k x^t$,

$$\mathbb{E}(f(\bar{x}^k) - f(x^*)) \leq \frac{4n}{k}\left[\frac{2L}{n}\|x^0 - x^*\|^2 + D_f(x^0, x^*)\right]$$

Stochastic gradient descent

Reduced variance method: SAGA

Rate for SAGA [Defazio, Bach, Lacoste-Julien, NeurIPS 2014]:

If the f_i's have L-Lipschitz gradient, then for $\tau = 1/(3L)$, one has, letting $\bar{x}^k := (1/k)\sum_{t=1}^k x^t$,

$$\mathbb{E}(f(\bar{x}^k) - f(x^*)) \le \frac{4n}{k}\left[\frac{2L}{n}\|x^0 - x^*\|^2 + D_f(x^0, x^*)\right]$$

- The method also allows for a prox-term $+\psi(x)$;
- Improved (linear) convergence rates if the f_i are μ-convex with L-Lipschitz gradient.
- (Older) variants such as "SVRG" re-compute $\nabla f(\bar{x})$ at some point $\bar{x}$ (which is also kept) from time to time, with the advantage that it is not needed to store all the v^i's as above. Then one can use $v_i = \nabla f_i(\bar{x})$ (recomputed when needed) and implement the same idea.

Example...

(see notebook)