Master 2 Data Science, Univ. Paris Saclay

Optimization for Data Science

Stéphane Gaïffas

Supervised learning setting

- features $a_i \in \mathbb{R}^d$
- labels $b_i \in \{-1, 1\}$ (binary classification), or
- labels $b_i \in \mathbb{R}$ (regression)

Loss function examples

- least-squares loss $\ell(b, b') = \frac{1}{2}(b - b')^2$ (linear regression)
- logistic loss $\ell(b, b') = \log(1 + e^{-bb'})$ (logistic regression)

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

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

For this kind of problems, you've seen so far

- Proximal gradient descent (GD), Accelerated proximal gradient descent (AGD)
- Coordinate descent (CD), coordinate gradient descent, (CGD)
- Linesearch methods (LS)
- Conjugate gradient (CG)
- Quasi-Newton methods: BFGS and L-BFGS

All the methods GD, AGD, CG, LS, BFGS and L-BFGS require the computation of

$$f(x) \quad \text{and} \quad \nabla f(x)$$

along iterations

Gradient descent uses iterations

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

and we know that if f is L-smooth then numerical complexity is

$$O(L/\varepsilon)$$

to achieve ε-precision, and if f is also μ-strongly convex then numerical complexity is

$$O\Big(\frac{L}{\mu}\log(1/\varepsilon)\Big)$$

to achieve ε-precision. We should say actually

$$O\Big(n\frac{L}{\mu}\log(1/\varepsilon)\Big)$$

if the "unit " is complexity of $\langle a_i, x\rangle$ ($O(d)$)

We say that these methods are based on **full gradients**, since at each iteration we need to compute

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

which depends on the whole dataset

Question. If n is large, computing $\nabla f(x)$ is long: need to pass on the whole data before doing a step towards the minimum!

Idea. Large datasets make your modern computer look old: go back to "old" algorithms.

Stochastic gradients

If I choose uniformly at random $I \in \{1, \ldots, n\}$, then

$$\mathbb{E}[\nabla f_I(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x)$$

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

Computation of $\nabla f_I(x)$ only requires the I-th line of data ($O(d)$ and smaller for sparse data, see next)

Stochastic Gradient Descent (SGD)

```
Input: starting point $x_0$, steps (learning rates) $\eta_t$
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 $x_t$
```

Remarks

- Each iteration has complexity $O(d)$ instead of $O(nd)$ for full gradient methods
- Possible to reduce this to $O(s)$ when features are s-sparse using **lazy-updates** (more on this later)

Full gradient descent

$$x_k \leftarrow x_{k-1} - \frac{\eta_k}{n} \sum_{i=1}^{n} \nabla f_i(x_{k-1})$$

has $O(nd)$ iteration: numerical complexity $O(n\frac{L}{\mu}\log(\frac{1}{\varepsilon})))$

Stochastic gradient descent

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

$O(d)$ iteration: numerical complexity $O(\frac{1}{\mu\varepsilon})$ (more next...)

(both when f is μ-strongly convex and L-smooth)

It does not depend on n for SGD !

Now x_t is a stochastic sequence, that depends on random draws of indices $i_1, \dots, i_t$, denoted $\mathcal{F}_t$

If i_t is chosen uniformly at random in $\{1, \dots, n\}$ and independent of previous $\mathcal{F}_{t-1}$ then

$$\mathbb{E}[\nabla f_i(x_{t-1}) | \mathcal{F}_{t-1}] = \frac{1}{n} \sum_{i'=1}^{n} \nabla f_{i'}(x_{t-1}) = \nabla f(x_{t-1})$$

SGD uses very noisy unbiased estimations of the full gradient

Polyak-Ruppert averaging: use SGD iterates x_t but return

$$\bar{x}_t = \frac{1}{t} \sum_{t'=1}^{t} x_{t'}$$

Theoretical properties on SGD. If:

- f is convex
- subgradients are bounded: $\|\nabla f_i(x)\|_2 \leq b$

we have a convergence rate

$$O\Big(\frac{1}{\sqrt{t}}\Big) \quad \text{with} \quad \eta_t = O\Big(\frac{1}{\sqrt{t}}\Big)$$

and if moreover

- f is μ-strongly convex

the rate is

$$O\Big(\frac{1}{\mu t}\Big) \quad \text{with} \quad \eta_t = O\Big(\frac{1}{\mu t}\Big)$$

Both achieved by ASGD (average SGD)

Under strong convexity, GD versus SGD is

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

GD leads to a more accurate solution, but what if n is very large?

Recipe
SGD is extremely fast in the early iterations (first two passes on the data)

But it fails to converge accurately to the minimum

Lazy updates.

Feature vectors are usually very sparse (bag-of-words, etc.). Complexity of the iteration can reduced from $O(d)$ to $O(s)$, where s is the sparsity of the features.

Typically $d \approx 10^7$ and $s \approx 10^3$

For minimizing

$$\frac{1}{n}\sum_{i=1}^{n}\ell(b_i, \langle x, a_i\rangle) + \frac{\lambda}{2}\|x\|_2^2$$

an iteration of SGD writes

$$x_t = (1 - \eta_t\lambda)x_{t-1} - \eta_t\ell'(b_i, \langle a_i, x_{t-1}\rangle)a_i$$

If a_i is s sparse, then computing $\eta_t\ell'(b_i, \langle a_i, x_{t-1}\rangle)a_i$ is $O(s)$, but $(1 - \eta_t\lambda)x_{t-1}$ is $O(d)$

Lazy updates trick.

Put $x_t = s_t\beta_t$, with $s_t \in [0,1]$ and $s_t = (1-\eta_t\lambda)s_{t-1}$

$$x_t = (1-\eta_t\lambda)x_{t-1} - \eta_t\ell'(b_i, \langle a_i, x_{t-1}\rangle)a_i$$

becomes

$$\begin{aligned} s_t\beta_t &= (1-\eta_t\lambda)s_{t-1}\beta_{t-1} - \eta_t\ell'(b_i, s_{t-1}\langle a_i, \beta_{t-1}\rangle)a_i \\ &= s_t\beta_{t-1} - \eta_t\ell'(b_i, s_{t-1}\langle a_i, \beta_{t-1}\rangle)a_i \end{aligned}$$

so the iteration is now

$$\beta_t = \beta_{t-1} - \frac{\eta_t}{s_t}\ell'(b_i, s_{t-1}\langle a_i, \beta_{t-1}\rangle)a_i$$

which has complexity $O(s)$.

[Convergence proofs for SGD on the blackboard]