Optimization for Machine Learning

November 2, 2022

Program of the next 4 sessions

- Stochastic gradient methods (Nov 2 + Nov 4)
- Large-scale optimization (Nov 7)
- Overview (Nov 10)

Stochastic gradient methods

↳ Many optimization problems in ML have the form

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Finite-sum structure

f_i depends on a data point from a dataset

Ex) Linear least squares

$X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$

$$f(w) = \frac{1}{2n} \|Xw - y\|^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} (x_i^T w - y_i)^2 \right)$$

↳ Typical setup: $n \gg 1$ ("big data")

Main cost of running an optimization algorithm: access the data

↳ Suppose that f_i is C^1 $\forall i = 1 \dots n$ (continuously differentiable)

Then f is also C^1, and we have $\forall w \in \mathbb{R}^d$,

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

$\Rightarrow$ One iteration of gradient descent

$$w_{k+1} = w_k - \alpha_k \nabla f(w_k)$$

involves computing a gradient of f

which implies computing all gradients

$\{\nabla f_i(w)\}_{i=1..n}$ $\Rightarrow$ Expensive when $n \gg 1$

($\approx$ looking at the entire dataset)

Goal: Find a method that is less expensive than gradient descent but still "works" in that it can converge to a solution of the optimization problem

Note: Here we consider $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ (empirical risk minimization)

but most of the reasoning applies to $f(w) = \mathbb{E}_\xi[f_\xi(w)]$ ξ sample following a data distribution

① Stochastic gradient (descent) SGD

pb: minimize $_{w \in \mathbb{R}^d}$ $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ f_i C^1 $\forall i=1..n$

Gradient descent iteration (GD)

$\forall k \in \mathbb{N}$, $w_{k+1} = w_k - \alpha_k \nabla f(w_k)$ $\alpha_k > 0$

Stochastic gradient iteration (SG)

$\forall k \in \mathbb{N}$, $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$ $\alpha_k > 0$

i_k randomly drawn in $\{1, \dots, n\}$

↳ One iteration of SG requires 1 access to an f_i (one access to a sample in the data set): this is cheaper than one iteration of GD that requires to access all samples!

Q) Does that method work?

→ Not always!

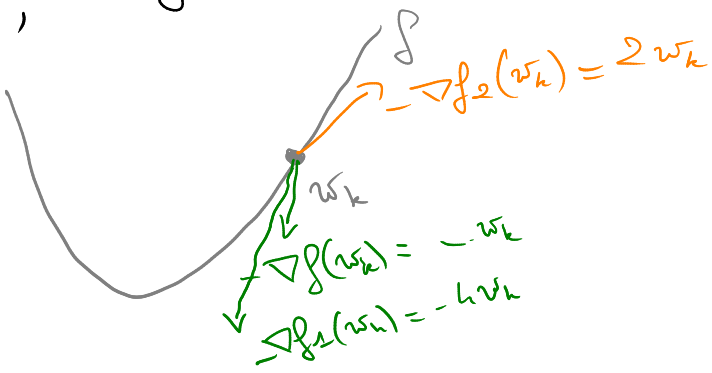
Ex) $n = 2$, $d = 1$, $f_1(w) = 2w^2$, $f_2(w) = -w^2$

$f(w) = \frac{1}{2}(f_1(w) + f_2(w))$

$= \frac{1}{2}w^2$

Suppose that you run SG on this problem and that you draw $i_k = 2$ at every k

Then, $\forall k$, $f(w_{k+1}) > f(w_k)$



⇒ SG is not guaranteed to converge because of the randomness at every iteration and the interplay between the f_i's and f

⇒ But in AI/ML practice, SG is typically much more efficient than GD!

② Implementation and analysis of SG

SG iteration: $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$

i_k randomly drawn in $\{1, \dots, n\}$

$\alpha_k > 0$

Q] How do you choose α_k? How do you draw i_k?]

Typical approaches for choosing α_k in SG: (stepsize / learning rate)

→ $\alpha_k = \alpha > 0$

⊕ Set a priori, no need for further calculation of α_k

⊖ The performance of SG can vary a lot according to the value of α ⇒ Learning rate tuning may be needed

→ $\{\alpha_k\}$ decreasing sequence (Ex: $\alpha_k = \frac{1}{\sqrt{k+1}}$)

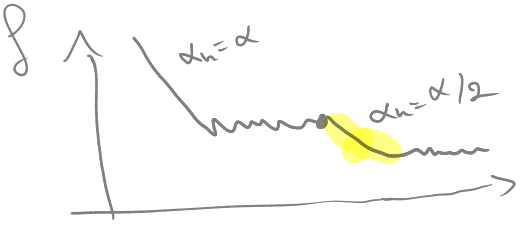
⊕ Prevents divergence

⊖ Can lead to slow convergence

→ Hybrid approach:

↳ start with $\alpha_k = \alpha > 0$ and run SG until the method "stalls" (e.g. the value of the true function $f(w_k)$ does not change much over a subset of successive iterations)

↳ Continue with $\alpha_k = \frac{\alpha}{2} < \alpha$ and repeat the process.



NB: Line-search techniques are not common in SG implementations because they require evaluations of the objective function f (and line searches over f_{i_k} do not work)

Assumptions on i_k

↳ We typically consider problems in which f is $C^{1,1}_L$ (C^1 + L-Lipschitz continuous gradient)

$$\Rightarrow \quad \forall\, (v,w) \in (\mathbb{R}^d)^2,$$

$$f(v) \leq f(w) + \nabla f(w)^T (v-w) + \frac{L}{2}\|v-w\|^2$$

The gradient descent step $v = w - \frac{1}{L}\nabla f(w)$ guarantees $f(v) \leq f(w) - \frac{1}{2L}\|\nabla f(w)\|^2$

⊕ Justifies the choice of $1/L$ as a stepsize

⊕ Guarantees convergence of GD

↳ Let $\{w_k\}$ be a sequence of iterates produced by stochastic gradient applied to $f \in C^{1,1}_L$.

Then, $\forall k \in \mathbb{N}$,

$$f(w_{k+1}) \leq f(w_k) + \nabla f(w_k)^T (w_{k+1} - w_k) + \frac{L}{2}\|w_{k+1} - w_k\|^2$$

$$= f(w_k) + \nabla f(w_k)^T (-\alpha_k \nabla f_{i_k}(w_k)) + \frac{L}{2}\|-\alpha_k \nabla f_{i_k}(w_k)\|^2$$

$$= f(w_k) - \alpha_k \nabla f(w_k)^T \nabla f_{i_k}(w_k) + \frac{L}{2}\alpha_k^2 \|\nabla f_{i_k}(w_k)\|^2$$

→ Because of the randomness in i_k, we cannot conclude from the inequality above that $f(w_{k+1}) < f(w_k)$ (which we could do in GD)

→ We can however provide guarantees on average by taking the expected value with respect to i_k in the inequality

⇒ Requires assumptions on i_k

Assumptions At every iteration, we assume that i_k is drawn independently from $i_0, \dots, i_{k-1}$ and α_k, and that $\nabla f_{i_k}(w_k)$ satisfies

(i) $\mathbb{E}_{i_k}[\nabla f_{i_k}(w_k)] = \nabla f(w_k)$ (expected value w.r.t. i_k)

(ii) $\mathbb{E}_{i_k}[\|\nabla f_{i_k}(w_k)\|^2] \leq \|\nabla f(w_k)\|^2 + \sigma^2$, $\sigma^2 \geq 0$

↳ $\mathbb{E}_{i_k}[\|\nabla f_{i_k}(w_k)\|^2] - \|\mathbb{E}_{i_k}[\nabla f_{i_k}(w_k)]\|^2 \leq \sigma^2$

On average, $\nabla f_{i_k}(w_k)$ reflects $\nabla f(w_k)$

The norm of the stochastic gradient $\nabla f_{i_k}(w_k)$ does not vary too much from the norm of $\nabla f(w_k)$

NB: Drawing i_k uniformly at random in $\{1, \dots, n\}$ satisfies these assumptions.

$$\mathbb{E}_{i_k}[\nabla f_{i_k}(w_k)] = \sum_{i=1}^{n} \mathbb{P}(i_k = i) \times \nabla f_i(w_k) = \sum_{i=1}^{n} \frac{1}{n} \times \nabla f_i(w_k) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_k) = \nabla f(w_k)$$

Proposition: Under these assumptions on f and $\{i_k\}$, for any $k \in \mathbb{N}$,

$$\mathbb{E}_{i_k}[f(w_{k+1})] \leq f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \frac{L}{2}\alpha_k^2 \|\nabla f(w_k)\|^2 + \frac{L\sigma^2\alpha_k^2}{2}$$

("Variance/Noise" term)

→ This result allows to define stepsize choices for which $\mathbb{E}_{i_k}[f(w_{k+1})] < f(w_k)$ (SG produces decrease in expectation)

→ Leads to guarantees after a number of iterations

Th: Guarantees with constant stepsize

Suppose that f is $C^{1,1}_L$ and μ-strongly convex.

Let w^* be the minimum (point) of f.

Suppose that we perform K iterations of SG with $\alpha_k = \alpha \leq 1/L$ and $\{i_k\}$ satisfies the assumptions above. Then,

$$\mathbb{E}\left[f(w_K) - f(w^*)\right] \leq (1-\mu\alpha)^K \left[f(w_0) - f(w^*) - \frac{L\alpha\sigma^2}{2\mu}\right] + \frac{L\alpha\sigma^2}{2\mu} \xrightarrow[K\to\infty]{} \frac{L\alpha\sigma^2}{2\mu}$$

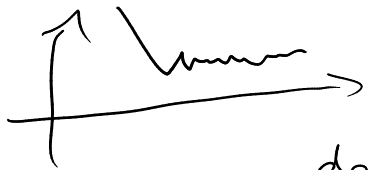
$\mathbb{E}$ = expectation with respect to $i_0, \dots, i_{K-1}$

↳ Recall: For GD with $\alpha_k = \alpha$, we would have

$$f(w_K) - f(w^*) \leq (1-\mu\alpha)^K (f(w_0) - f(w^*)) \xrightarrow[K\to\infty]{} 0$$

- GD guarantees that $f(w_K) \xrightarrow[K\to\infty]{} f(w^*)$ deterministically
- SG guarantees in expectation that $\mathbb{E}[f(w_K)] \xrightarrow[K\to\infty]{} [f(w^*), f(w^*) + \frac{L\alpha\sigma^2}{2\mu}]$

→ weaker guarantee
→ reflects the practical behavior



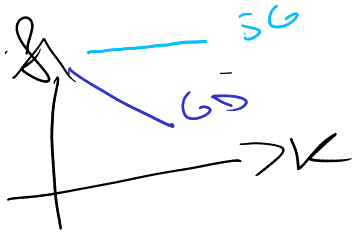
(NB: in practice we do of the order of SG iterations, which partly explains why we observe this behavior)

Th) Under the assumptions of the previous theorem, suppose now that we run $K \geq 1$ iterations of SG with decreasing stepsize $\alpha_k = O\left(\frac{1}{k+1}\right)$

then, $\mathbb{E}\left[f(w_K) - f(w^*)\right] \leq O\left(\frac{1}{K}\right)$

↳ GD: $f(w_K) - f(w^*) \leq O\left(\left(1 - \frac{\mu}{L}\right)^K\right)$

. $\left(1 - \frac{\mu}{L}\right)^K$ goes to 0 more quickly than $\frac{1}{K}$ ⇒ GD has a better CV rate than SG!



For the same number of iterations (K), the final value $f(w_K)$ of the GD iterates is closer to $f(w^*)$ than the final value for SG

→ BUT comparing SG and GD in terms of iterations is not the right metric if we are concerned about accesses to data points

⇒ A better metric is based on the notion of an epoch.

Def: An epoch is a unit that corresponds to accessing n times a point in a dataset of size n (n times the same pt, 1 time every pt, ...)

Key observation:

→ 1 iteration of GD $\nabla f(w_k) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_k)$ accesses all data points and it costs 1 epoch

→ 1 iteration of SG accesses 1 data point and it costs $1/n$ epoch

1 epoch $\equiv$ 1 iteration of GD $\equiv n$ iteration of SG

Going back to the last theorem, suppose that we run GD and SG with a budget of M epochs:

↳ for GD: $\mathcal{O}\left(\left(1-\frac{\mu}{L}\right)^M\right)$

↳ for SG: $\mathcal{O}\left(\frac{1}{nM}\right)$

and $\frac{1}{nM} << \left(1-\frac{\mu}{L}\right)^M$ when $n >> 1$

(usually, M is smaller than n)

In terms of epochs, the convergence rates of SG are (on average/in expectation) better than that of GD!