Session 13: Stochastic gradient descent

Optimization and Computational Linear Algebra for Data Science

Léo Miolane

Final exam

- Scope: everything except today's lecture and this week's video.
- Of course, it will be a bit more focused on what we did after the midterm (PCA, linear regression, convex functions, optimization...)
- Same format as for the midterm
- "24 hours window" on Thursday December 17th.
- 1 hour 40 minutes to work + 20 minutes to scan + upload on Gradescope.
- In case you have any issue when uploading: **email me your work**.

Contents

Introduction

Supervised learning

Assume we have a pair of random variables (X, Y) that is distributed according to some distribution P_0.

Example: X = MRI image of a brain

$$Y = \begin{cases} +1 & \text{if there is a cancer} \\ 0 & \text{otherwise.} \end{cases}$$

Goal: Predict Y from X. using a model

φ_θ

models' parameter.

Hope: $\varphi_\theta(X) \simeq Y$

Supervised learning

Find θ that minimizes the risk:

$$\underline{R(\theta)} = \mathbb{E}\left[\ell\left(\varphi_\theta(X), Y \right) \right]$$

expectation wrt $(X,Y) \sim \underline{P_0}$

loss function, for instance $\ell(a,b) = (a-b)^2$

ISSUE: we can not compute $R(\theta)$ because we do not know P_0.

However, we may have access to samples

$$\underbrace{(X_1, Y_1), \dots (X_N, Y_N) \overset{iid}{\sim} P_0}_{\text{"dataset"}}.$$

minimize the empirical risk:

$$\boxed{R_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(\varphi_\theta(X_i), Y_i\right)}$$

Supervised learning

$$R_N(\theta) = \frac{1}{N} \sum_{i=1}^{\boxed{N}} \underbrace{\ell(\varphi_\theta(x_i), y_i)}_{f_i(\theta)}$$

"loss on the i^{th} sample only"

Example: Linear regression

$$\ell(a,b) = (a-b)^2$$

$$\varphi_\theta(x) = \langle \theta, x \rangle$$

$$R_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\langle \theta, x_i \rangle - y_i \right)^2$$

$$= \frac{1}{N} \| X\theta - y \|^2$$

$$X = \begin{pmatrix} - x_1 - \\ \vdots \\ - x_N - \end{pmatrix}$$

Why not using gradient descent ?

$$R_N(\theta) = f(\theta) = \frac{1}{N}\sum_{i=1}^{N} f_i(\theta).$$

$\theta \in \mathbb{R}^n$

Gradient descent iterations:

$$\theta_{t+1} = \theta_t - \alpha_t \overbrace{\nabla f(\theta_t)}$$

$$= \theta_t - \frac{\alpha_t}{N}\sum_{i=1}^{N} \underbrace{\nabla f_i(\theta_t)}_{\in \mathbb{R}^n}.$$

Computing $\nabla f_i(\theta_t)$ takes $\simeq$ n operations.

Computing $\nabla f(\theta_t)$ takes $\simeq$ $\underline{N \cdot n}$ operations.

too large if we have a lot of data.

Stochastic gradient descent

Stochastic gradient descent

$$f(\theta) = \frac{1}{N} \sum_{i=1}^{N} f_i(\theta).$$

Starting at some $\theta_0 \in \mathbb{R}^n$, perform the updates:

Pick i uniformly at random in $\{1, \ldots, N\}$,

Update $\theta_{t+1} = \theta_t - \alpha_t \nabla f_i(\theta_t)$,

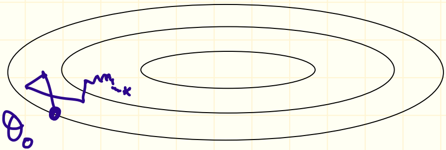
→ takes only n computations per iterations.

$\nabla f_i(\theta_t)$ can be seen as a noisy version of $\nabla f(\theta_t)$

SGD :
$$\theta_{t+1} = \underbrace{\theta_t - \alpha_t \nabla f(\theta_t)}_{\text{Gradient descent}} + \underbrace{\alpha_t \text{"Noise"}}_{\text{noise term.}}$$

Tradeoffs in SGD

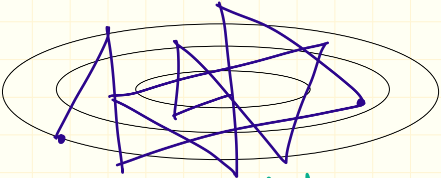
Rapidly decaying step sizes



⊕ Kills the noise

⊖ We may "never forget the initial condition"

Slowly decaying step sizes



⊕ We move fast.

⊖ We may oscillate too much.

It is usually recomendended that

- $\alpha_t \xrightarrow[t \to +\infty]{} 0$ "Kill the noise"
- $\sum_{t=0}^{+\infty} \alpha_t = +\infty$ "Forget the initial condition"

- $\alpha_t = (0,9)^t \xrightarrow[t \to +\infty]{} 0$

$$\sum_{t=0}^{+\infty} \alpha_t = \frac{1}{1-0,9} \quad \text{finite.}$$

- $\alpha_t = \frac{1}{t} \xrightarrow[t \to +\infty]{} 0$

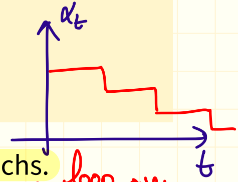
$$\sum_{t=1}^{+\infty} \frac{1}{t} = +\infty$$

SGD in practice

Mini-batch stochastic gradient descent: ← reduces the noise

Pick a mini-batch $i_1, \ldots, i_k$ in $\{1, \ldots, N\}$,

Update $$\theta_{t+1} = \theta_t - \frac{\alpha_t}{k} \sum_{m=1}^{k} \nabla f_{i_m}(\theta_t),$$



- Decrease the step size after a fixed number of epochs. loop over the data set
- Use momentum + "adaptive gradient": Adagrad, RMSprop, Adedelta, Adam, Adamax, Nadam...

Excellent reference:

`https://arxiv.org/pdf/1609.04747.pdf`

Convergence analysis

Convergence rates

- if the f_i are convex and L-smooth: SGD with $\alpha_t = 1/\sqrt{t}$ achieves an error $\leq C/\sqrt{t}$.
- if the f_i are μ-strongly convex and L-smooth: SGD with $\alpha_t = 1/(\mu t)$ achieves an error $\leq C/t$.

Compare with gradient descent :

- L smooth $\longrightarrow \leq \frac{\text{Constant}}{t}$
- L-smooth + μ- strongly cvx $\longrightarrow \leq \text{Constant} \cdot \left(1 - \frac{\mu}{L}\right)^t \bigg) \leq e^{-\frac{\mu}{L} t}$.

GD vs SGD

Gradient descent

- Time per step $n \cdot N$

nb. of param. → n

nb. of samples → N

- Error after t steps

$$\leq C \cdot e^{-\rho t}$$

$\rho \to \frac{\mu}{L}$

(strongly convex case)

- Log-error after τ units of time

$$-\rho \frac{\tau}{N \cdot n} + \text{cste}.$$

Stochastic gradient descent

- Time per step n.

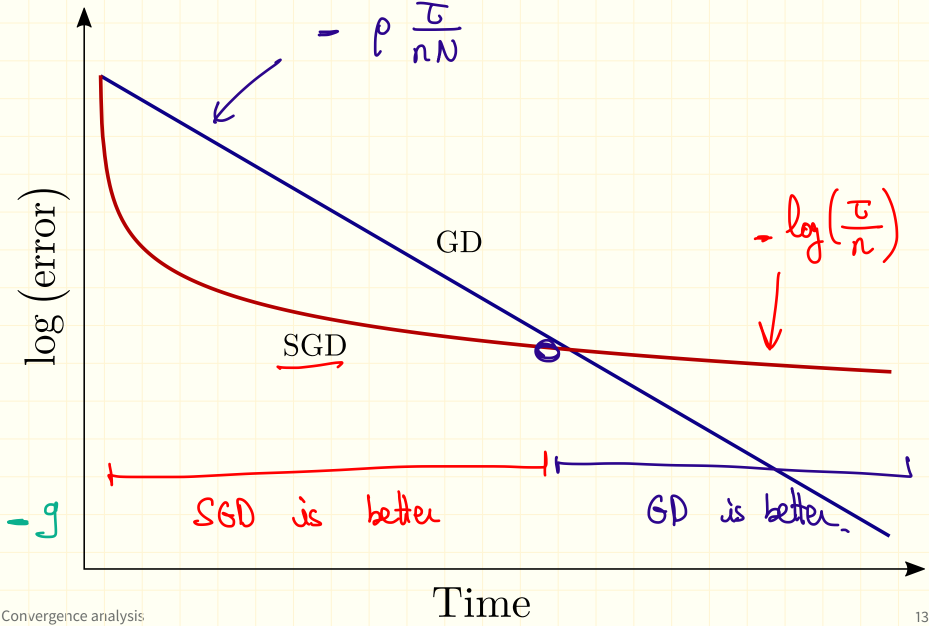
- Error after t steps

$$\leq \frac{C}{t}$$

- Log-error after τ units of time

$$-\log\left(\frac{\tau}{n}\right) + \text{cste}.$$

GD vs SGD


$-\rho \frac{\tau}{nN}$
GD
SGD
$-\log\left(\frac{\tau}{n}\right)$
log (error)
Time
$-g$
SGD is better
GD is better.

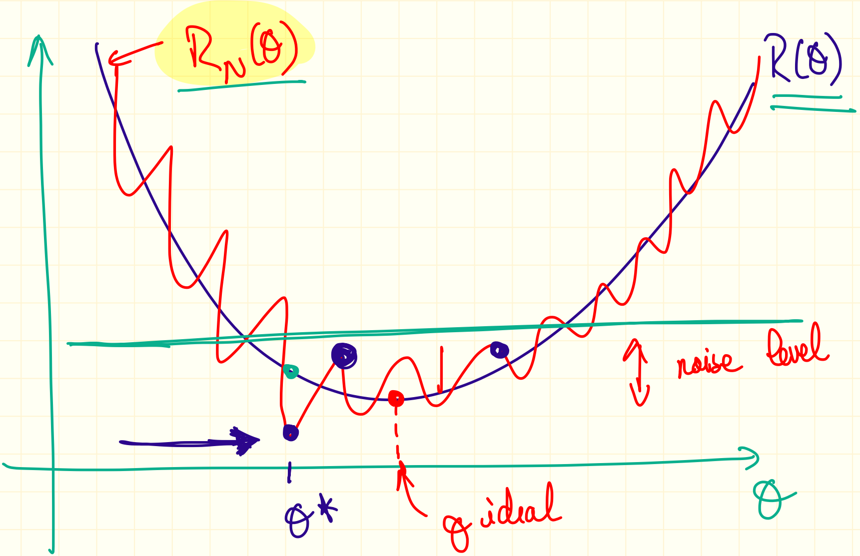
GD vs SGD: who wins ?

- If one is looking for a very small optimization error $f(\theta_t) - \min f$, then gradient descent wins.
- If one has a limited time budget and does not need a very small $f(\theta_t) - \min f$, then stochastic gradient descent wins.

IN machine learning, it makes no sense to optimize "below the noise level"

→ SGD is a popular choice.

Questions?



Questions?

$$(u_t) : \begin{cases} \overset{X^2}{u_{t+2}} = \underbrace{u_{t+1}}_{X} - 2\,\underbrace{u_t}_{1}. \\ u_0 = 1 \\ u_1 = 1 \end{cases}$$

Solve: $\boxed{X^2 = X - 2}$

$\boxed{X^2 - X + 2 = 0}$: $\Delta = 1 - 8 = -7$

solutions: $X_+ = \dfrac{1 + i\sqrt{7}}{2}$ $\quad X_- = \dfrac{1 - i\sqrt{7}}{2}$

Questions?

Then

$$u_t = a(X_+)^t + b(X_-)^t$$

constants that needs to be determined using the initial conditions.

$$|u_t| \leqslant C \cdot \left(|X_+|^t + |X_-|^t\right) \leqslant \underline{C' \sqrt{2}^t}$$

$$|X_+| = \left|\frac{1 + i\sqrt{7}}{2}\right| = \sqrt{\frac{1}{4} + \frac{7}{4}} = \sqrt{2}$$

$$|X_-| = \sqrt{2}$$

Questions?

$$\begin{pmatrix} u_{t+2} \\ u_{t+1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -2 \\ 1 & 0 \end{pmatrix}}_{M} \cdot \begin{pmatrix} u_{t+1} \\ u_t \end{pmatrix}$$

$$u_{t+2} = u_{t+1} - 2u_t$$

$$0 = \det(M - \lambda \mathrm{Id}) = \det \begin{pmatrix} 1-\lambda & -2 \\ 1 & -\lambda \end{pmatrix}$$

$$= -\lambda(1-\lambda) + 2 = \boxed{\lambda^2 - \lambda + 2} = 0$$

$$|u_{t+2}| \leq \sqrt{u_{t+2}^2 + \underbrace{u_{t+1}^2}_{\geq 0}} \leq \underset{2C}{2C} \cdot \left(\lambda_1^{t+2} + \lambda_2^{t+2}\right)$$