Optimization for machine learning – Lectures II & IV
Stochastic gradient descent

Rodolphe Le Riche[1], Didier Rullière[2]

[1] CNRS LIMOS, Mines Saint-Etienne, UCA, France
[2] Mines Saint-Etienne, CNRS LIMOS

Dec 2025
Majeure Science des données (UP3) et Master Maths en Action
Mines Saint-Etienne

Course program I

Lecture I. (Rodolphe). Non stochastic optimization for ML

- Introduction
  Objectives – Optimization problem formulation – Examples – Basic mathematical concepts for optimization
- Steepest descent algorithm
  Fixed step steepest descent algorithm – Line search – convergence
- Improved gradient based searches
  Search directions for acceleration – Restarts – Constraints

Lectures II. (Didier). Stochastic optimization for ML

- Stochastic Approximation SA, Stochastic Gradient Descent SGD. This lecture ☺
  Robbins-Monro

Lecture III. (Rodolphe). Non stochastic Gradient calculation by backpropagation

- Towards ML: regularized quadratic function
- Application to neural network, retropropagation

Lecture IV. (Didier). Stochastic optimization for ML

- Unknown gradient. Neural Applications and Batches.
  Kiefer-Wolfowitz – Applications to ML

Course program II

... Future lectures about optimization

- Global optimization
  Using metamodels – EGO – CMAES – Simulated Annealing... ← (UP4, upcoming)

Gradient descent - following the steepest descent ...

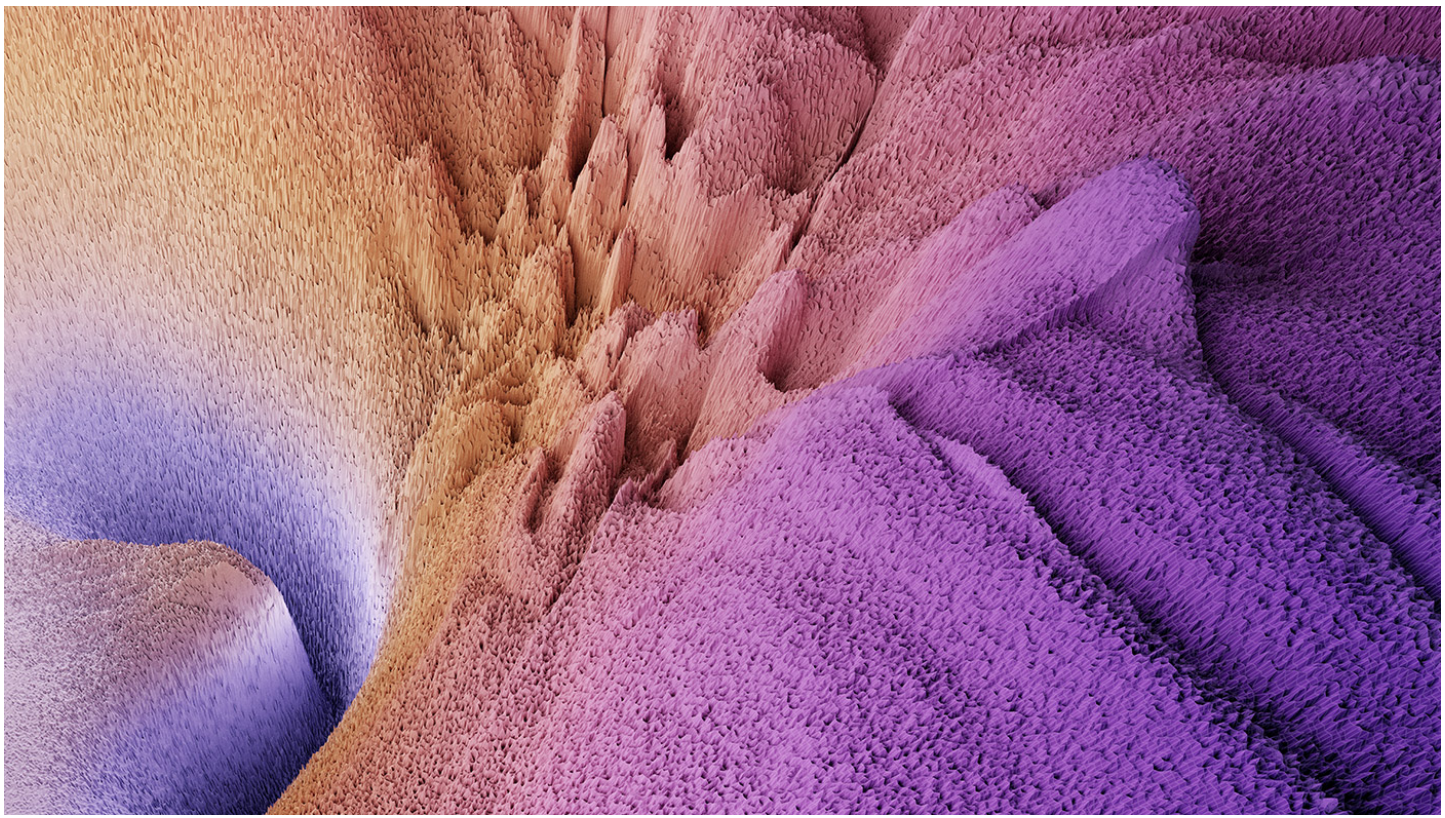
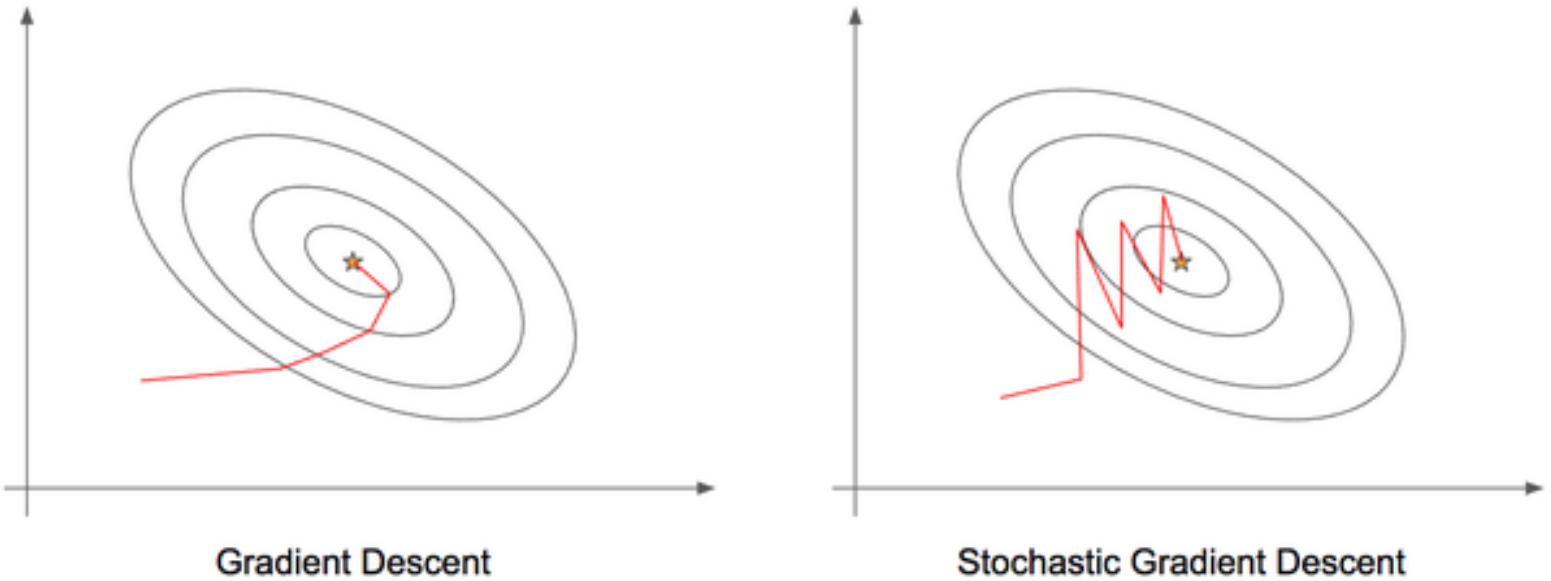


image from https://losslandscape.com/

Steepest descent: *Cauchy et al.*, 1847, historical review *Petrova and Solov'ev*, 1997.

Stochastic Gradient Descent



images from
https://pythonmachinelearning.pro/complete-guide-to-deep-neural-networks-part-2/
front page image from https://summer.pes.edu/programs/page/2/

Stochastic gradient descent

1. Motivation

Stochastic gradient descent - Why?

A **Stochastic Gradient Descent** is for optimizing a noisy function.

... but ...

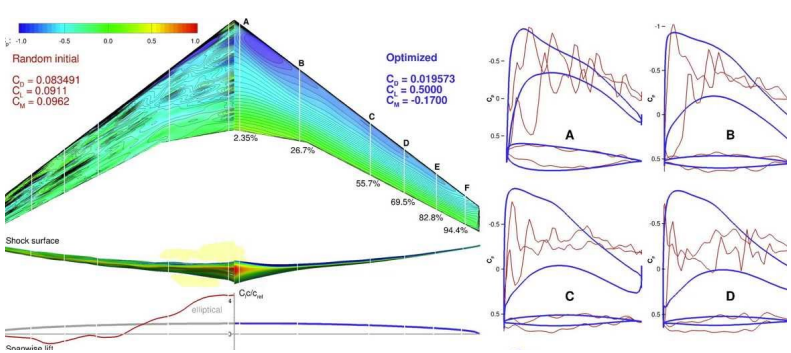
Why optimizing a **noisy** function ? where is the noise?

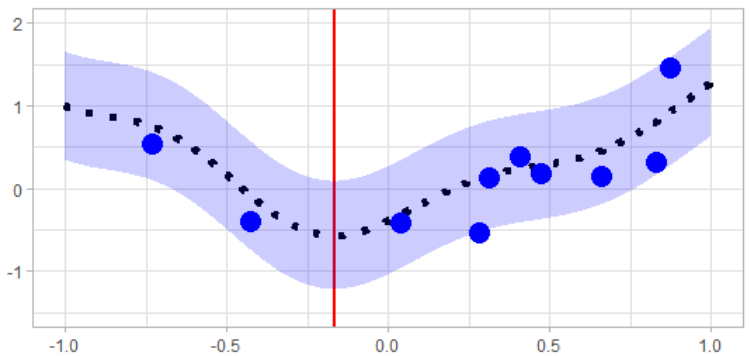
Stochastic gradient descent - Why?

Application example 1: noise in experiment measures

You want to optimize $f(\theta)$, but due to noise in experiments you only observes

$$F(\theta) = f(\theta) + noise(\theta)$$





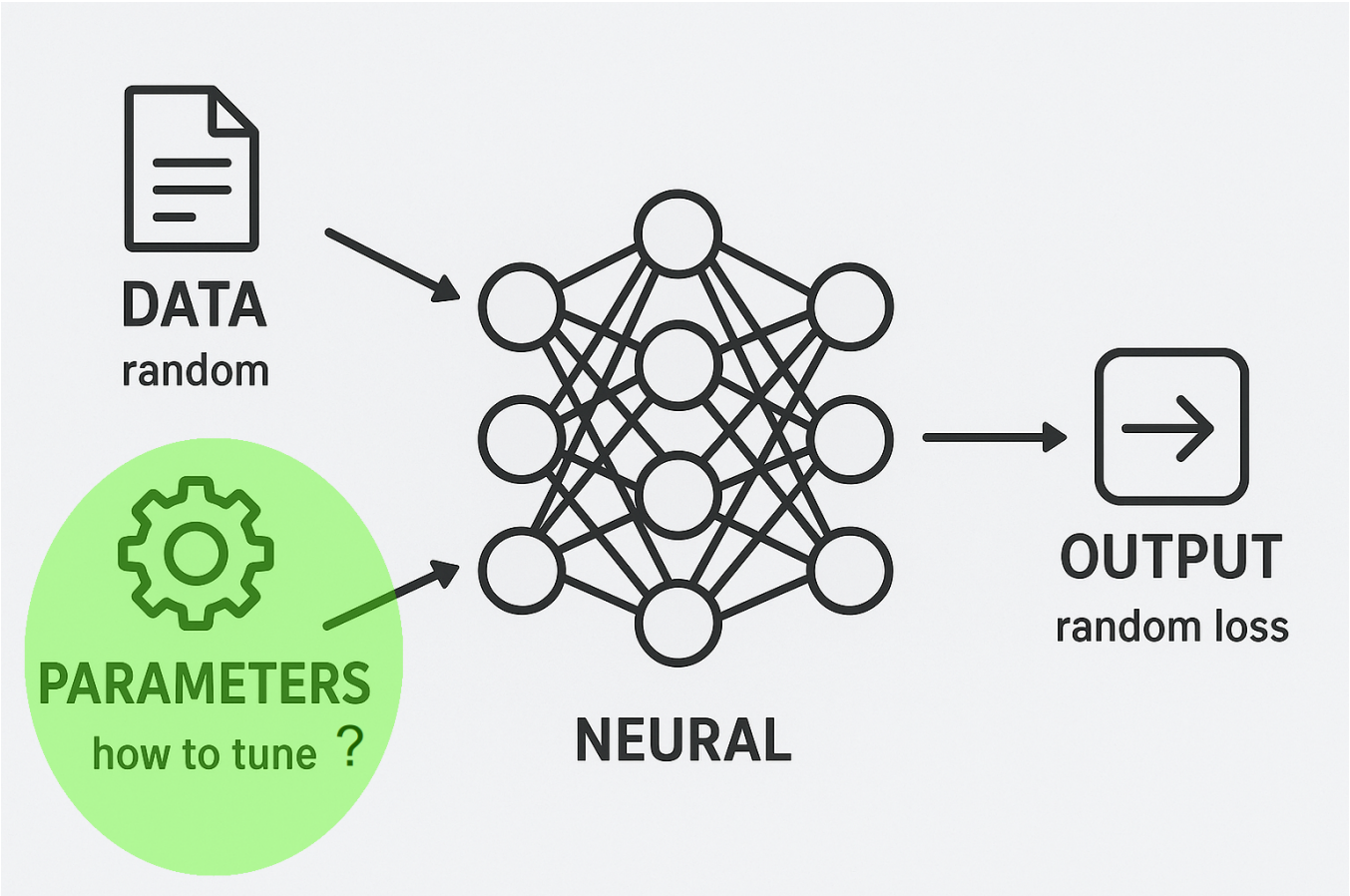
left image from `https://youtu.be/nuC-3X7Uxmc`

(fictive situation)

- Minimize wing friction $f(\theta)$ (dotted line), as a function of a shape parameter θ.
- Experiments in wind tunnel lead only to noisy measurements $F(\theta)$ (blue dots)

Stochastic gradient descent - Why?

Application example 2: machine learning parameter optimization



Stochastic gradient descent - Why?

Application example 2: machine learning parameter optimization (details)

- Choose a model depending on a vector of parameters, say $\boldsymbol{\theta}$
- Find the best model, i.e. the best parameter $\boldsymbol{\theta}^\star$ ☺

The prediction error for a parameter $\boldsymbol{\theta}$ is measured by a loss function:

$$\ell(\boldsymbol{\theta}, \mathrm{Data}).$$

This is random! it depends on random (sampled, noisy) data.

Thus one needs to minimize a deterministic function

$$f(\boldsymbol{\theta}) := \mathrm{E}\left[\ell(\boldsymbol{\theta}, \mathrm{Data})\right]$$

given only observed random loss at some parameters $\boldsymbol{\theta}$

$$F(\boldsymbol{\theta}) := \ell(\boldsymbol{\theta}, \mathrm{Data}).$$

Stochastic gradient descent - Why?

Application examples: summary

Finally, in both cases, one needs to find

$$\theta^\star \in \operatorname*{argmin}_{\theta} \mathrm{E}\left[F(\theta)\right] \tag{1}$$

given only noisy (and costly) observations $F(\theta) = f(\theta) + noise(\theta)$.

"*The stochasticity might come from the evaluation at random subsamples (minibatches) of datapoints, or arise from inherent function noise*"

Kingma and Ba, 2014

Stochastic gradient descent - Why?

Useful: deeply used in deep Neural Networks and LLM !

chatGPT answer to "*detail in three points if stochastic gradient descent is important for optimizing chatGPT*":

Importance of Stochastic Gradient Descent for Optimizing ChatGPT

- **Efficiency and Speed:** Stochastic Gradient Descent (SGD) speeds up training by using mini-batches instead of the entire dataset, allowing for faster convergence.
- **Improved Generalization:** The randomness in SGD helps the model escape local minima, enhancing its ability to generalize well on unseen data.
- **Scalability:** SGD effectively handles large datasets, making it scalable for training ChatGPT on diverse linguistic inputs without excessive computational resources.

SGD is **crucial** for optimizing ChatGPT, balancing efficiency, generalization, and scalability.

Optimizing a noisy function - How?

How to minimize a function observed with noise?

- Many possible stochastic optimizers (see *Fouskakis and Draper*, 2002): local search, genetic algorithms, simulated annealing, ...
- We focus here on:

see `https://www.youtube.com/watch?v=Anc2_mnb3V8`, visual comparison local search vs SGD

Optimizing a noisy function - How?

Aim of the lecture

- Presenting a minimal setting to understand sto. grad. descent
- Giving main ideas of some proofs
- Explaining the conditions needed
- Making theory more modular (separating SGD, Neural Net)
- Discussing the impact of step sizes

What we do not develop here

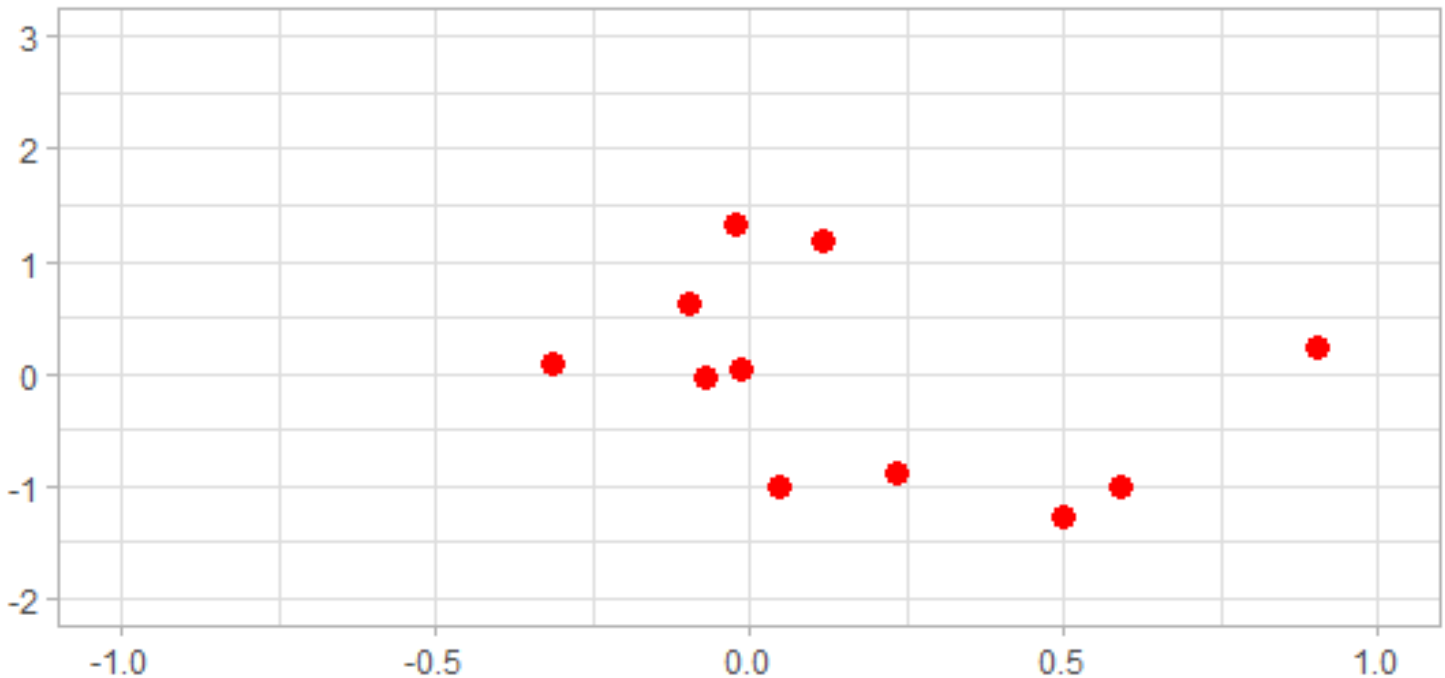
- Recent refinements of the theory
- Specific settings (e.g. constraints)
- Ready-to-use specific libraries

Remarks

- Main notations are given in Appendix ☺
- Please use the forum on *Campus* if you have questions

Optimizing a noisy function - quick idea (I)

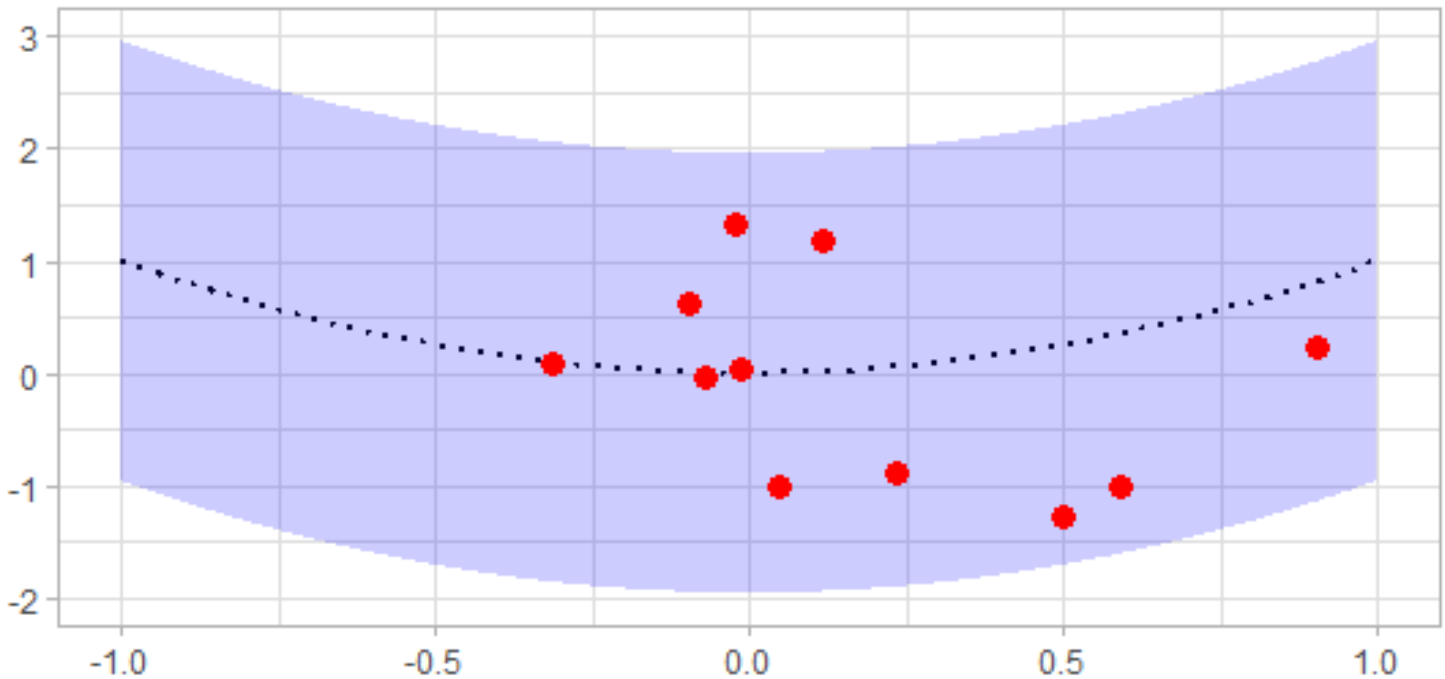
Optimizing a noisy function



One observes: a function + a noise, at some points. How to find the minimum ?

Optimizing a noisy function - quick idea (II)

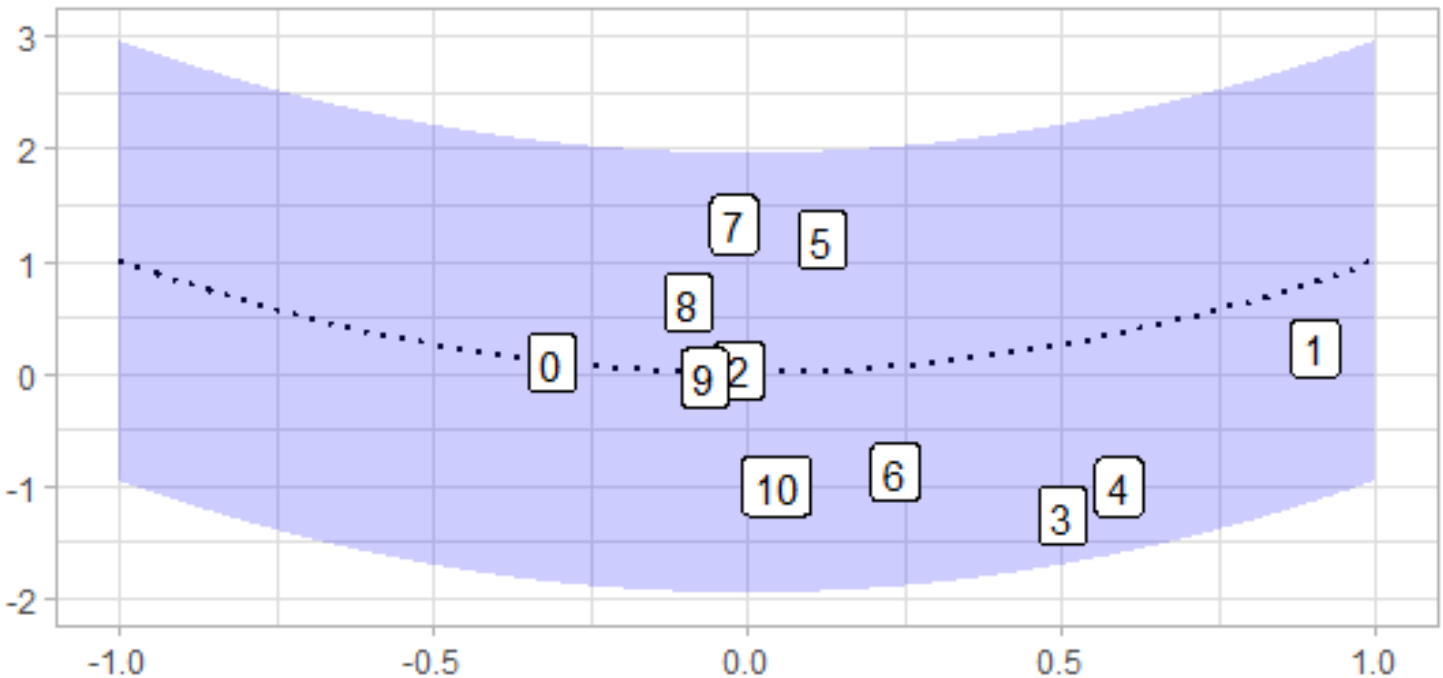
Optimizing a noisy function



It always looks easier when one shows the hidden reality behind :-)

Optimizing a noisy function - quick idea (III)

Optimizing a noisy function



Here is an illustration of a real stochastic gradient descent... How to do this?

2. Stochastic Approximation: finding roots

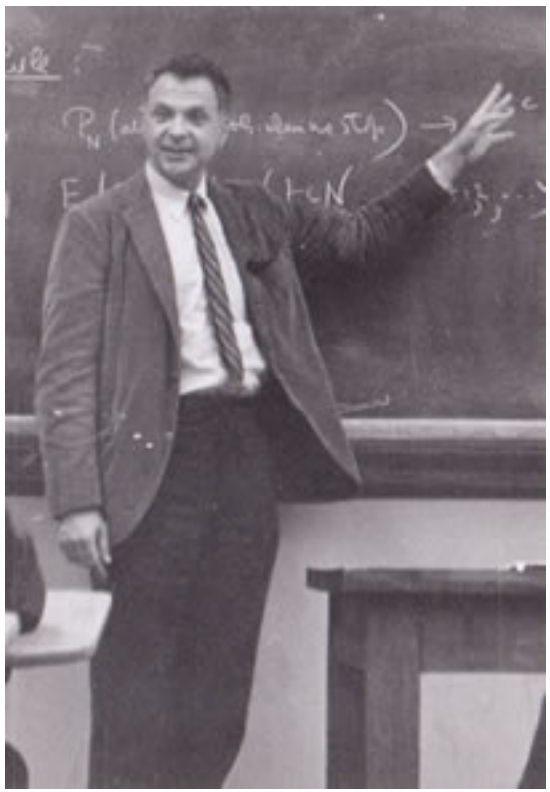
Stochastic approximation

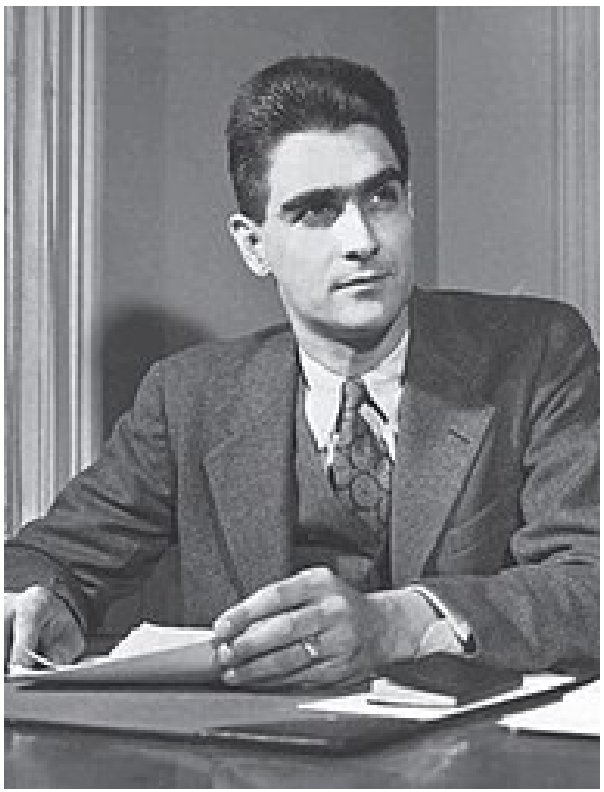
Let us start with finding the root of a noisy function.

this is called:

Stochastic Approximation (SA)

Stochastic approximation - 1D model





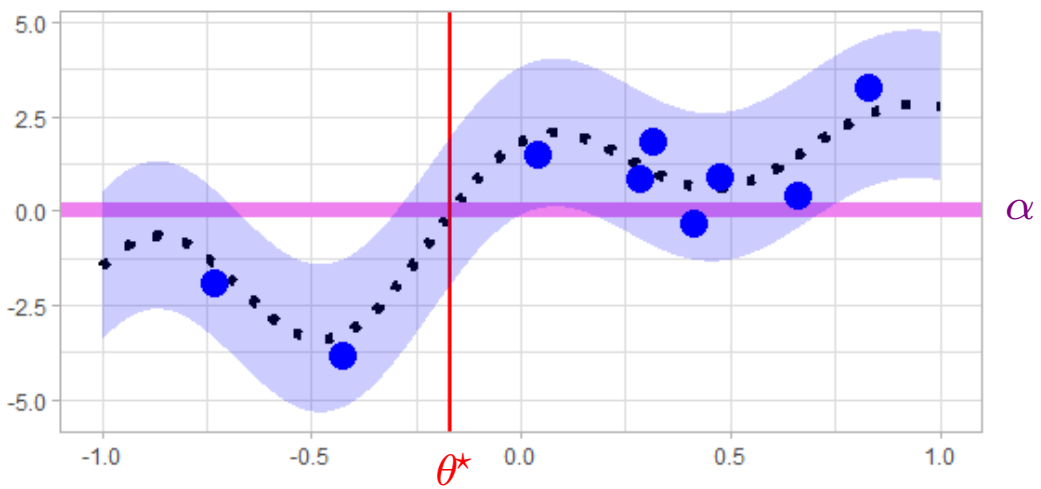
Herbert Robbins and Sutton Monro

Stochastic approximation - 1D model

Let us start with finding the root of a noisy function.

- $g(.)$ is an unknown function (black dotted line)
- $g(.)$ has a unique root $\theta^\star$ such that $g(\theta^\star) = \alpha$, we want it (red vertical line)
- $g(.)$ is first below then above the threshold α (violet threshold)
- One only observes real random variables $G(\theta) = g(\theta) + \mathcal{E}_\theta$ (blue dots)

...other conditions coming...



g (dotted line) is first below then above the threshold. Blue dots are $G(\theta_1)...G(\theta_n)$.

Stochastic approximation - 1D model

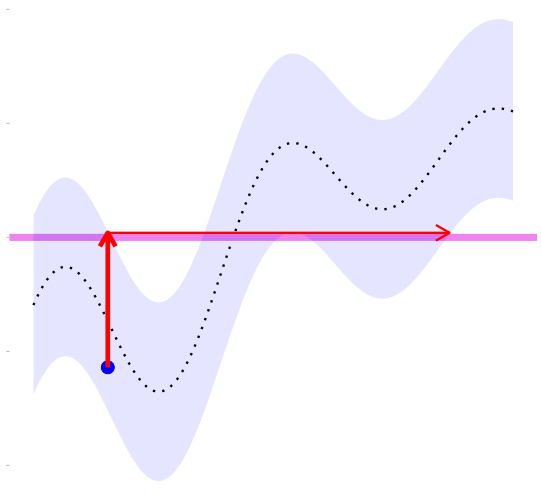
Robbins-Monro Stochastic approximation (SA)

Finding root in θ of $g(\theta) = \mathrm{E}[G(\theta)] = \alpha$, when only noisy $G(\theta) = g(\theta) + \mathcal{E}_\theta$ are observed, with $\mathrm{E}[\mathcal{E}_\theta] = 0$, *Robbins and Monro, 1951*:

$$\Theta_{n+1} = \Theta_n + a_n (\alpha - G(\Theta_n)) \tag{2}$$

starting from $\Theta_1 = \theta_1 \in \mathbb{R}$, with given sequence $\{a_n\}$.

- a_n step size in the direction of crossing.
- α known value of the target



$\Longrightarrow$ increases Θ_{n+1} if $G(\Theta_n) < \alpha$, decreases otherwise.

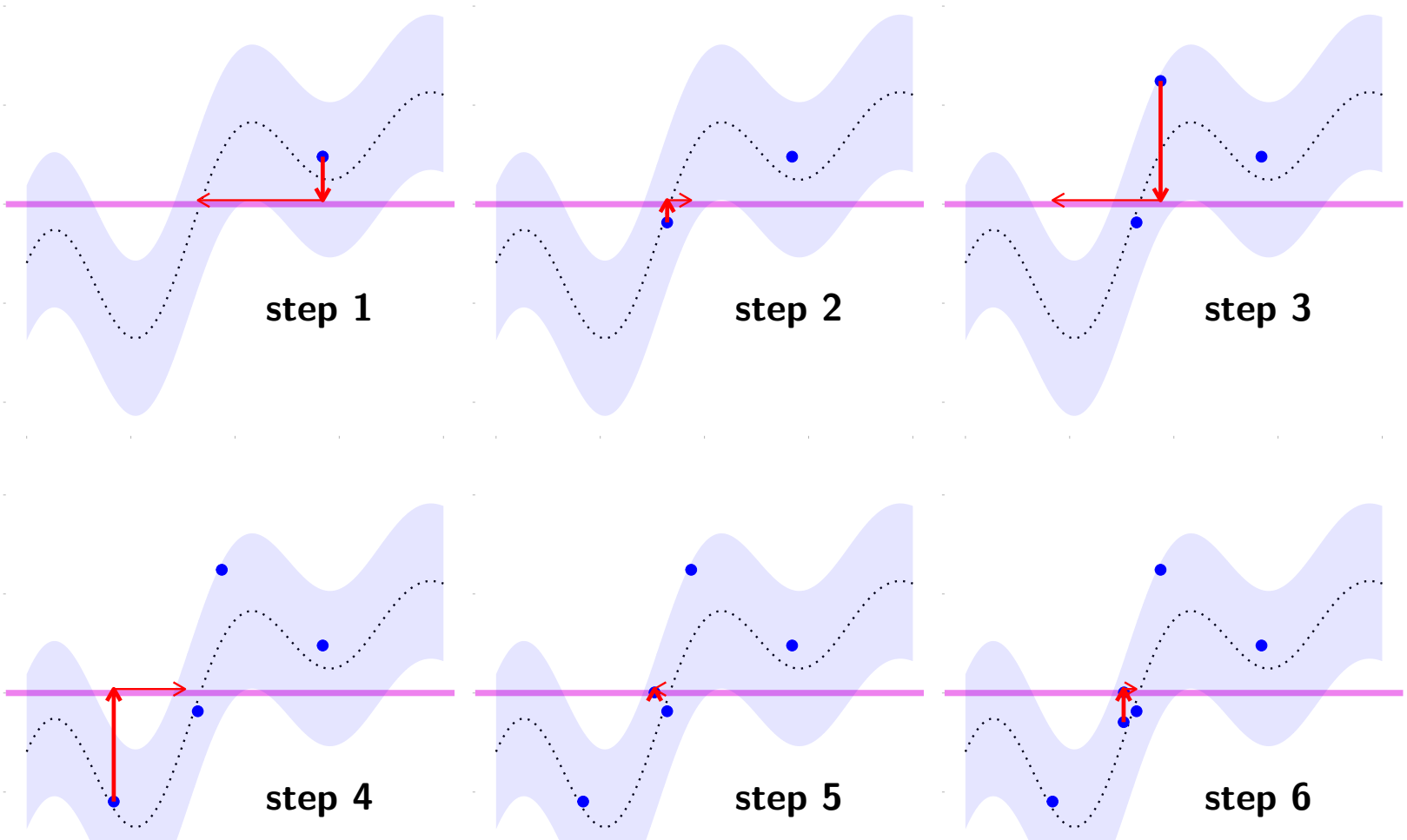
further iterations

Stochastic approximation - 1D model

An illustration of the recursion $\Theta_{n+1} = \Theta_n + a_n (\alpha - G(\Theta_n))$

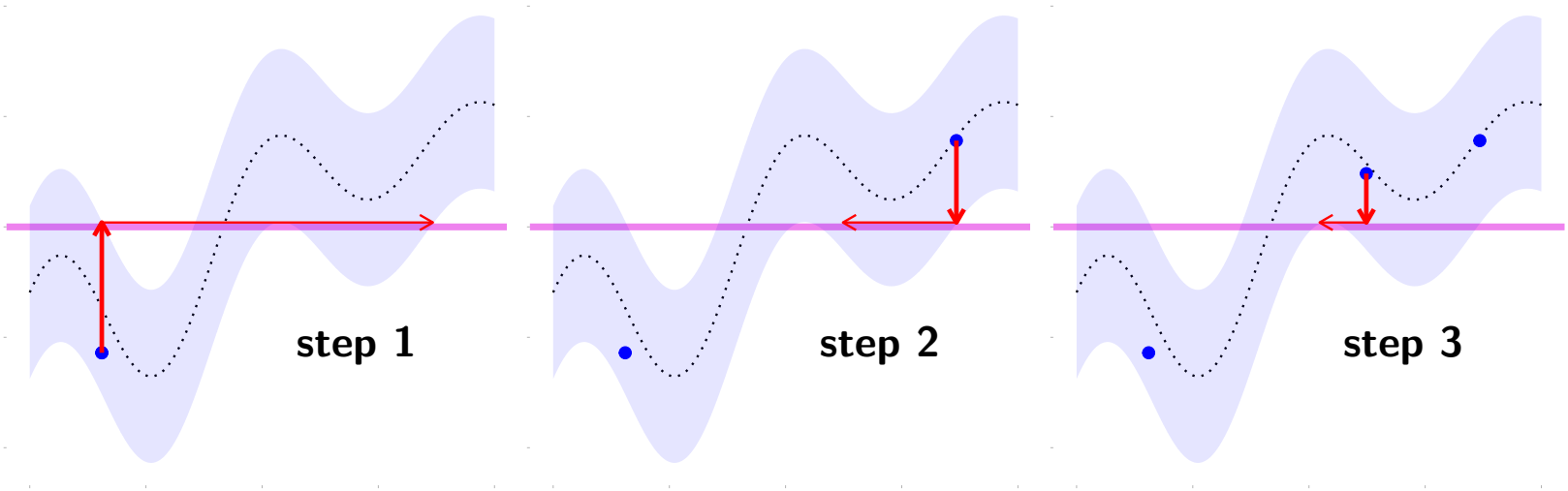
▸ Other seeds

recall g (dotted line) is first below then above the threshold. Blue dots are $G(\Theta_n)$.



Stochastic approximation - 1D model

recall g (dotted line) is first below then above the threshold. Blue dots are $G(\Theta_n)$.



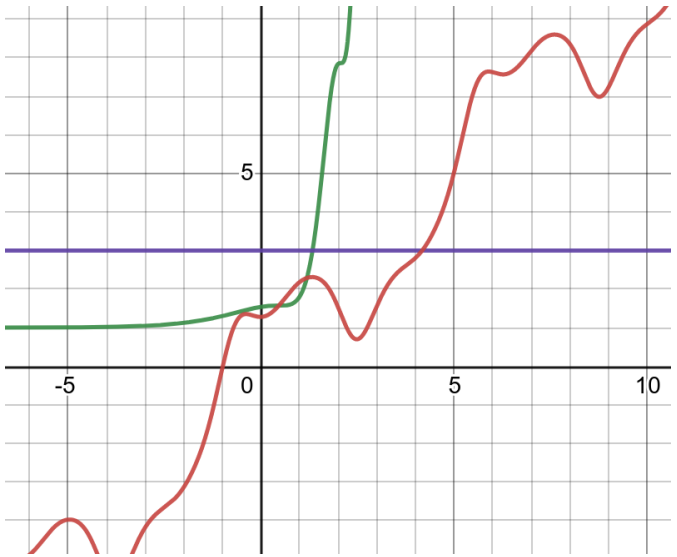
QUIZZ Q1: intuitively, which conditions on g and noisy observations G?

QUIZZ Q2: intuitively, which conditions on $\{a_n\}$?

Stochastic approximation - Q1: Conditions on G

Conditions on $G(.)$ and $g(.)$, *Wolfowitz, 1952*.

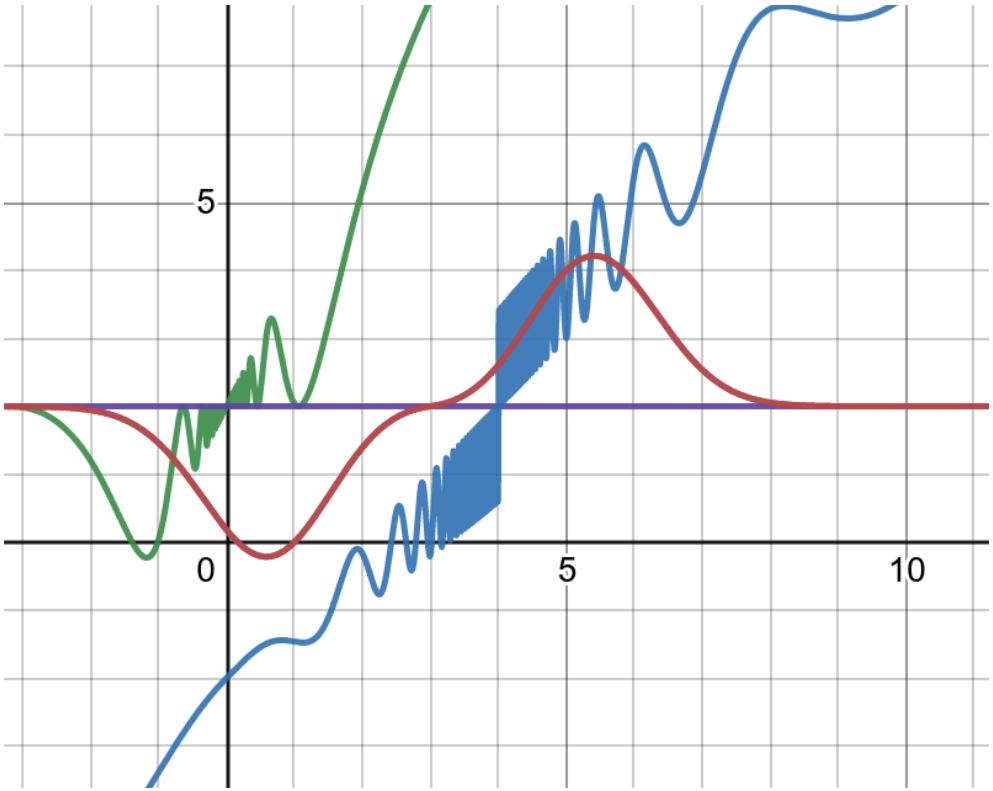
- **c0**: $G(.)$ has bounded mean and variance.
  i.e. $|g(\theta)| \leq C < \infty$, $\mathrm{Var}\left[G(\theta)\right] < \infty$
- **c1**: $g(.)$ below then above threshold α.
  i.e. $g(\theta) < \alpha$ for $\theta < \theta^\star$, $g(\theta) > \alpha$ for $\theta > \theta^\star$
- **c2**: $g(.)$ strictly increasing near $\theta^\star$.
  i.e. $\exists \delta > 0, g(\theta)$ st. increasing if $|\theta - \theta^\star| < \delta$
- **c3**: $|g(\theta) - \alpha|$ not too small far from $\theta^\star$.
  i.e. $\inf_{|\theta - \theta^\star| \geq \delta} |g(\theta) - \alpha| > 0$



The user has usually no choice on the shape of G,
$\Longrightarrow$ conditions help understanding failures.

Stochastic approximation - Q1: Conditions on G

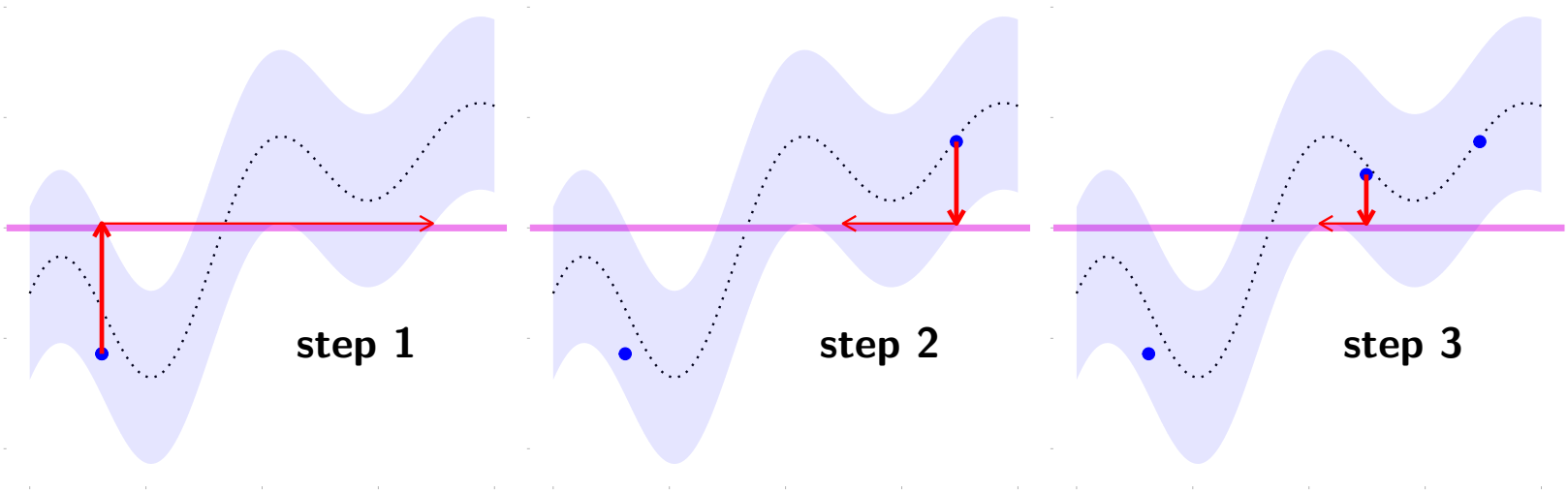
An example of pathological functions $g(.)$ that do not meet the conditions **c2** or **c3**



QUIZZ What would happen in such cases?

Stochastic approximation - Q2: Conditions on a_n

remind the illustration of the recursion $\Theta_{n+1} = \Theta_n + a_n (\alpha - G(\Theta_n))$:



Conditions on positive coefficients $\{a_n\}$?

- **h1**: $\sum a_n = +\infty$
- **h2**: $\sum a_n^2 < +\infty$
- $a_n \to 0$ (implicit)

The user chooses these coefficients, huge importance on the convergence!

▸ Skip convergence proof

Stochastic approximation - Idea of the proof (I)

Step 0: Main quantity to study:

$$\mathrm{E}\left[(\Theta_n - \theta^\star)^2\right] .$$

Applying the recursion (2), $\Theta_{n+1} = \Theta_n + a_n(\alpha - G(\Theta_n))$, conditioning on Θ_n, we get

$$\mathrm{E}\left[(\Theta_{n+1} - \theta^\star)^2\right] = \mathrm{E}\left[(\Theta_n - \theta^\star)^2\right] - 2a_n \mathrm{E}\left[(\Theta_n - \theta^\star)(g(\Theta_n) - \alpha)\right] + a_n^2 \mathrm{E}\left[(G(\Theta_n) - \alpha)^2\right] .$$

Hence by summation

$$\mathrm{E}\left[\left(\Theta_{n+1} - \theta^\star\right)^2\right] = \mathrm{E}\left[\left(\theta_1 - \theta^\star\right)^2\right] - 2\sum_{j=1}^{n} a_j \mathrm{E}\left[(\Theta_j - \theta^\star)(g(\Theta_j) - \alpha)\right] + \sum_{j=1}^{n} a_j^2 \mathrm{E}\left[(G(\Theta_j) - \alpha)^2\right] .$$

... does it converge?

Stochastic approximation - Idea of the proof (II)

We had

$$\mathrm{E}\left[\left(\Theta_{n+1}-\theta^\star\right)^2\right] = \mathrm{E}\left[\left(\theta_1-\theta^\star\right)^2\right] - 2\sum_{j=1}^{n} a_j \,\mathrm{E}\left[(\Theta_j-\theta^\star)(g(\Theta_j)-\alpha)\right] + \sum_{j=1}^{n} a_j^2 \,\mathrm{E}\left[(G(\Theta_j)-\alpha)^2\right]$$

Step 1: let us bound each sum.

- By **c0**: finite mean and variance, $\mathrm{E}\left[(G(\Theta_j)-\alpha)^2\right]$ bounded

  By **h2**: $\sum a_j^2 < +\infty$, (too high a_j would create noise explosion)

$$\sum_{j=1}^{\infty} a_j^2 \,\mathrm{E}\left[(G(\Theta_j)-\alpha)^2\right] < +\infty$$

- By **c1**: $g(.)$ below then above threshold α,

$$(\Theta_j-\theta^\star)(g(\Theta_j)-\alpha) = |\Theta_j-\theta^\star|\,|g(\Theta_j)-\alpha| \geq 0$$

As $\mathrm{E}\left[(\Theta_{n+1}-\theta^\star)^2\right] > 0$ and $\sum_{j=1}^{\infty} a_j^2 \,\mathrm{E}\left[(G(\Theta_j)-\alpha)^2\right] < +\infty$ bounded, we get

$$\sum_{j=1}^{\infty} a_j \,\mathrm{E}\left[|\Theta_j-\theta^\star|\,|g(\Theta_j)-\alpha|\right] < +\infty \tag{3}$$

Stochastic approximation - Idea of the proof (III)

we had

$$\sum_{j=1}^{n} a_j \, \mathrm{E}\left[|\Theta_j - \theta^\star| \, |g(\Theta_j) - \alpha| \right] < +\infty \tag{3}$$

Step 2: Use above finite absolute sum

- By **h1**: $\sum a_j = +\infty$, hence (too small a_j would not force convergence)

$$\liminf_{n \to +\infty} \mathrm{E}\left[|\Theta_j - \theta^\star| \, |g(\Theta_j) - \alpha| \right] = 0 \tag{4}$$

  - either $|\Theta_j - \theta^\star|$ becomes small.
  - or either $|g(\Theta_j) - \alpha|$ becomes small, excluded far from $\theta^\star$ by **c2** and **c3**.
- with skipped details (see *Wolfowitz, 1952*), one shows

$$\Theta_n \underset{n \to +\infty}{\overset{P}{\to}} \theta^\star \tag{5}$$

Stochastic approximation - Idea of the proof (IV)

more details on the step 2 (facultative):

$$\liminf_{n \to +\infty} \mathrm{E}\left[|\Theta_j - \theta^\star| \, |g(\Theta_j) - \alpha| \right] = 0$$

- there exists a subsequence of indices $n_1 < n_2 < \dots$ so that the limit is 0.
- $\Theta_{n_j} \xrightarrow{P} \theta^\star$: otherwise $\exists \epsilon > 0$, $\eta > 0$ and a subsubsequence $t_1 < t_2 < \dots$ such that

$$\mathrm{P}\left[\left| \Theta_{t_j} - \theta^\star \right| > \eta \right] > \epsilon \quad \text{is impossible:}$$

by **c2**, and **c3** $|g(\theta) - \alpha|$ not too small far from $\theta^\star$, would lead to contradiction

$$\mathrm{E}\left[\underbrace{\left| \Theta_{t_j} - \theta^\star \right|}_{A} \underbrace{\left| g(\Theta_{t_j}) - \alpha \right|}_{B} \right] \geq \epsilon \, \eta \inf_{|\theta - \theta^\star| > \eta} |g(\theta) - \alpha|$$

because $A \geq 0, B \geq 0$, hence

$$\mathrm{E}[AB] \geq \underbrace{\mathrm{P}[A > \eta]}_{> \epsilon} \underbrace{\mathrm{E}[AB \,|\, A > \eta]}_{\geq \eta \inf \dots}$$

- with skipped details (see *Wolfowitz, 1952*), one shows

$$\Theta_n \underset{n \to +\infty}{\xrightarrow{P}} \theta^\star$$

3. Stochastic Gradient Descent

Stochastic Gradient Descent

Let us continue with finding the minimizer of a noisy function.

this is here:

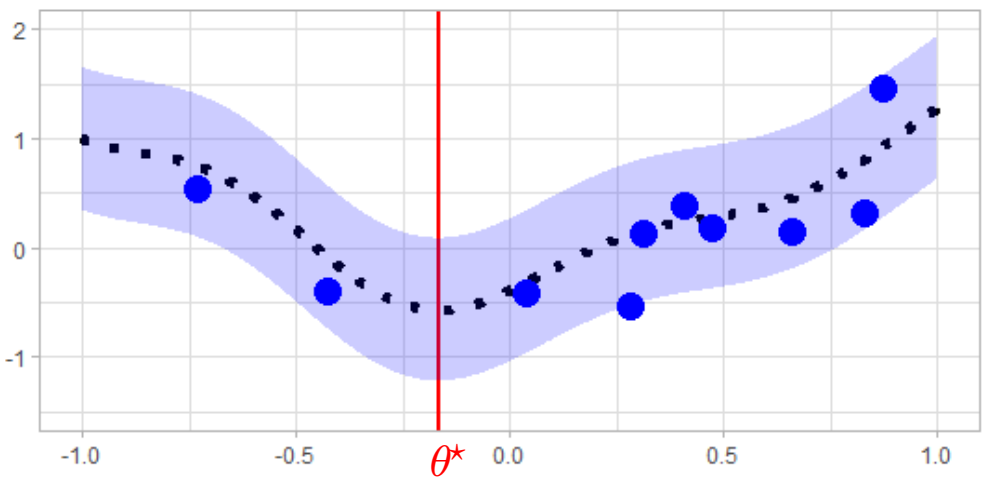
Stochastic Gradient Descent (SGD)

Stochastic gradient descent - 1D model

Let us continue with finding the minimizer of a noisy function.

- $f(.)$ is an unknown function (black dotted line)
- $f(.)$ has a unique minimizer $\theta^\star$, we want to find it. (red vertical line)
- One observes real random variables $F(\theta) = f(\theta) + \mathcal{E}_\theta$ (blue dots)

...other conditions coming...



f (dotted line) has a unique minimizer. Blue dots are $F(\theta_1)...F(\theta_n)$.

Stochastic gradient descent - 1D model

Adaptation to gradient descent with known noisy gradient

- Noisy observations $F(\theta)$ and $f(\theta) = \mathrm{E}[F(\theta)]$. ($F$ and f not needed).
- Known noisy derivative $\widehat{G}(\theta)$, such that $\mathrm{E}\left[\widehat{G}(\theta)\right] = \frac{\partial}{\partial\theta} f(\theta)$.
- One applies Stochastic Approximation to $\widehat{G}(.)$ and threshold $\alpha = 0$.
- One assumes $\widehat{G}(.)$ satisfies Robbins-Monro assumptions, as $G(.)$ previously.

Robbins-Monro adaptation to Stochastic Optimization - 1D (SGD)

Starting from $\Theta_1 = \theta_1$, the sequence

$$\Theta_{n+1} = \Theta_n - a_n \widehat{G}(\Theta_n) \tag{6}$$

converges to $\theta^\star = \mathrm{argmin}_\theta f(\theta)$ in probability, or almost surely under more conditions.

Can be directly adapted

- to maximization, changing $-a_n$ in $+a_n$.
- to several dimensions, replacing derivative by gradient.

Stochastic gradient descent - 1D model example

An example on estimating a mean

Let $f(\theta) = \frac{1}{2}\,\mathrm{E}\left[(N-\theta)^2\right]$ for a r.v. N, say $N \sim \mathcal{N}(\mu, \sigma^2)$. It is minimal in

$$\theta^\star = \mathrm{E}\left[N\right].$$

QUIZZ Show that this $\theta^\star$ is the minimizer. Hint: $N - \theta = (N - \mathrm{E}\left[N\right]) - (\theta - \mathrm{E}\left[N\right])$.

Maximum Likelihood Estimator

Given an iid sequence $N_1, \ldots, N_n$ distributed as N, define the ML estimator of $\mathrm{E}\left[N\right]$

$$\widehat{\theta^\star_n} = \frac{1}{n}(N_1 + \ldots + N_n)\,.$$

Its variance decreases rapidly, in $1/n$: $\mathrm{Var}\left[\widehat{\theta^\star_n}\right] = \frac{\sigma^2}{n}$.

Robbins-Monro like

$\widehat{G}(\theta) = (\theta - N_n)$ is a noisy derivative: $\mathrm{E}\left[\widehat{G}(\theta)\right] = \frac{d}{d\theta} f(\theta)$.

$$\Theta_{n+1} = \Theta_n - \mathsf{a}_n \cdot (\Theta_n - N_n)\,,$$

QUIZZ Show that, when $\mathsf{a}_n = \frac{1}{n}$ and $\Theta_1 = 0$, one has $\Theta_{n+1} = \widehat{\theta^\star_n}$, excellent estimator!

QUIZZ Just remark that $f''(\theta^\star) = 1$ and $\mathsf{a}_n = \frac{1}{n f''(\theta^\star)}$ in the previous example...

Stochastic gradient descent - Multivariate

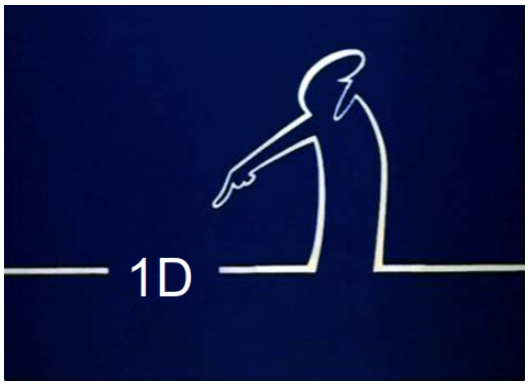


image adapted from Osvaldo Cavandoli *La Linea*

Adaptation to gradient descent with known noisy gradient

- Noisy observations $F(\theta)$, and $f(\theta) = \mathrm{E}\left[F(\theta)\right]$.
- Known noisy gradient $\widehat{G}(\theta)$, such that $\mathrm{E}\left[\widehat{G}(\theta)\right] = \nabla f(\theta)$.

Robbins-Monro adaptation to Stochastic Optimization - Multivariate (SGD)

Starting from $\mathbf{\Theta}_1 = \theta_1$, the sequence

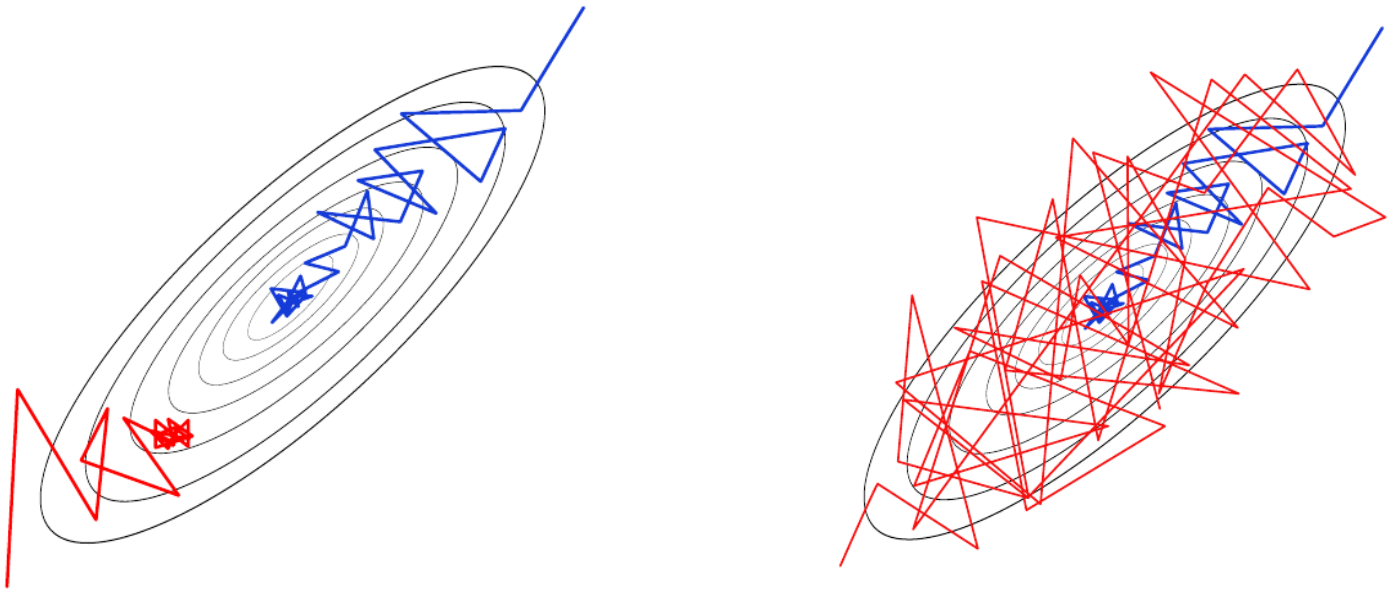
$$\mathbf{\Theta}_{n+1} = \mathbf{\Theta}_n - a_n \widehat{G}(\mathbf{\Theta}_n) \tag{7}$$

converges to $\theta^\star = \mathrm{argmin}_\theta f(\theta)$ almost surely under *Blum*, 1954 conditions.

QUIZZ What is the impact of the step sizes $\{a_n\}$?

Impact of the step sizes

Too low or too high values for $\{a_n\}$



images from https://www.cs.ubc.ca/~nando/550-2008/lectures/Mar19.pdf

Recall the bias-variance trade-off:

$$\underbrace{\mathrm{E}\left[(\Theta_n - \theta^\star)^2\right]}_{\text{error (MSE)}} = \underbrace{(\mathrm{E}\left[\Theta_n\right] - \theta^\star)^2}_{\text{bias}} + \underbrace{\mathrm{Var}\left[\Theta_n\right]}_{\text{variance}}$$

QUIZZ For red paths, which situation has high bias? high variance? what if $a_n = 0$?

Stochastic gradient descent - convergence intuition (I)

Understanding convergence, the "noise ball"

Let us consider the case of **constant step sizes**: $\forall n,\ a_n = a$. The SGD writes

$$\Theta_{n+1} = \Theta_n - a\widehat{G}(\Theta_n) \tag{8}$$

One can show that there exists a constant $v_{\text{ball}} > 0$ such that (hidden assumptions)

$$\lim_{n \to +\infty} \mathrm{E}\left[\|\Theta_n - \theta^\star\|^2\right] = v_{\text{ball}} > 0 \tag{9}$$

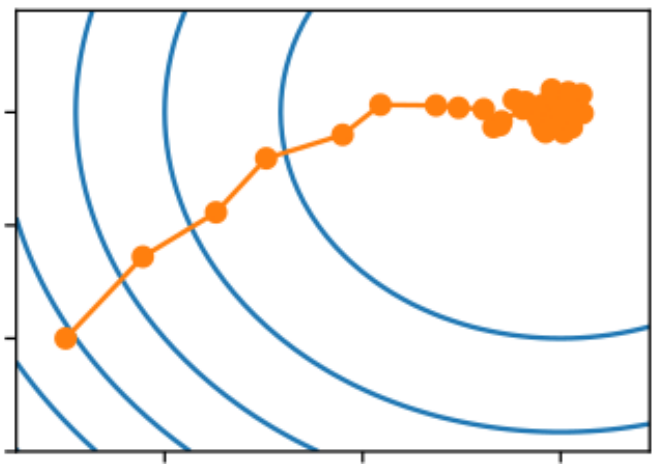


illustration from https://d2l.ai/chapter_optimization/sgd.html

QUIZZ What happens if we choose a smaller value of a? a larger value? conclude...

Stochastic gradient descent - convergence intuition (II)

Understanding the noise ball with a minimal RM code in R

$G : \theta \mapsto 2\theta + \epsilon$ is a noisy gradient of $\theta \mapsto \theta^2$ and a is a constant(!) step.

```
G <- function(theta) {2*theta + rnorm(1, mean=0, sd=1)}
a <- 0.3
Theta <- vector(length=100)
Theta[1] <- 2
for(n in 1:99) {
  Theta[n+1] <- Theta[n] - a * G(Theta[n])
}
plot(Theta, type="l", col="blue")
```

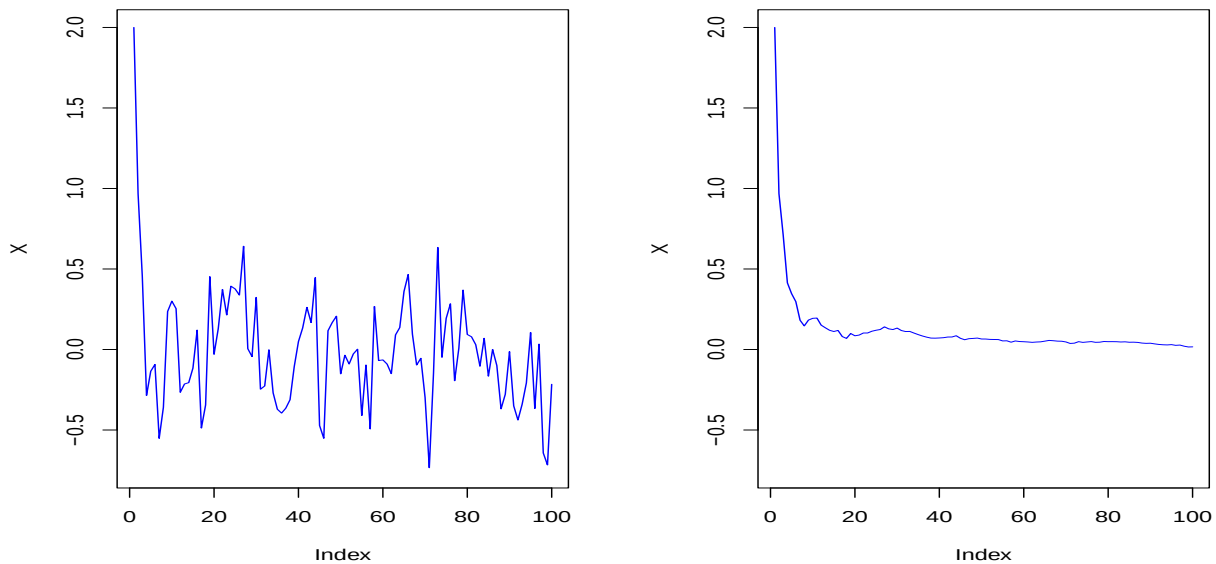


Figure on the right: a was replaced by a/n in the loop of the algo (red arrow).

Stochastic gradient descent - convergence intuition (III)

impact of (constant) step size a = "learning rates" in machine learning

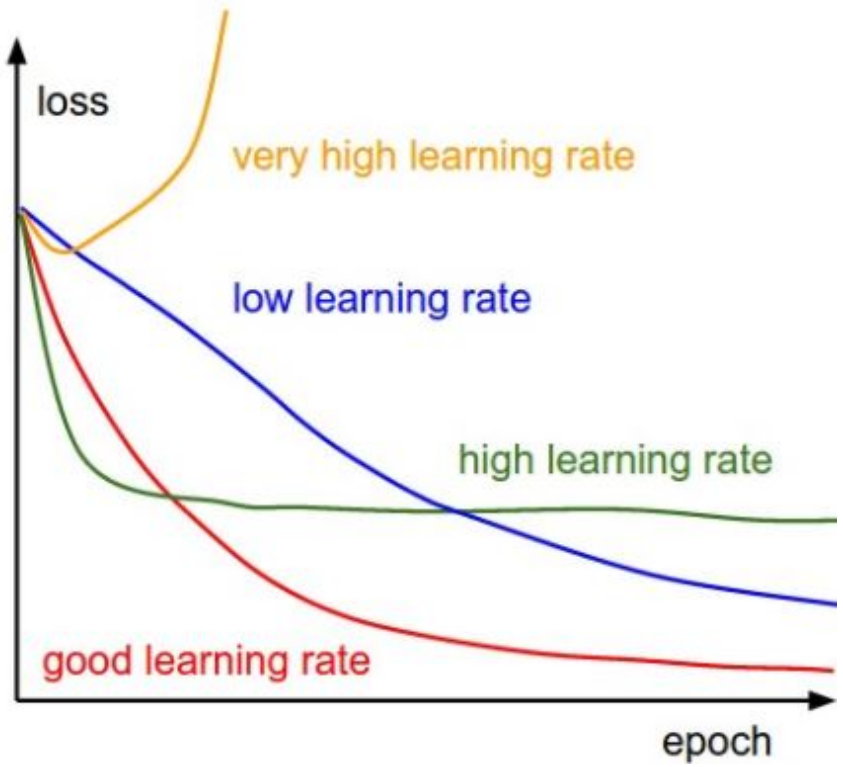


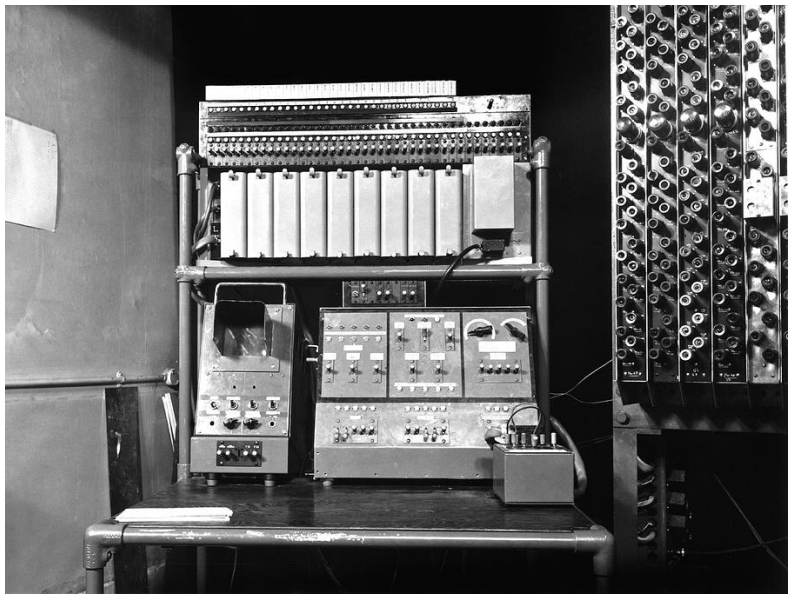
image from
https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788397872/1/ch01lvl1sec19/forward-and-backpropagation

QUIZZ Illustrate the very high learning rate with a parabola optimization.

Stochastic gradient descent

Historical remark

Robbins Monro article is published in 1951...



A computer in 1950 ☺

... this was before Neural Networks implementations ...

Stochastic gradient descent - Abscissa averaging (I)

Few remarks on the convergence

- When $\mathsf{a}_n = \mathsf{a}/n^\alpha$, $\alpha \in (1/2, 1]$.
- When $\mathsf{a}_n = \mathsf{a}/n^\alpha$, in 1D-model, best theoretical asymptotic variance is obtained when (see *Ruppert, 1988*, adapt to SGD). Possible multivariate extensions.

$$\mathsf{a}_n = \frac{1}{f''(\theta^\star) n}$$

$f''(\theta^\star)$ is unknown... can be estimated iteratively (adaptative procedures).

- When $\mathsf{a}_n = \mathsf{a}/n^\alpha$, one can use lower values of α if one do (abscissa) averaging:

$$\overline{\boldsymbol{\Theta}}_n = \frac{1}{n_0} \sum_{j=n-n_0+1}^{n} \boldsymbol{\Theta}_j$$

for a given lag n_0, $1 \leq n_0 \leq n$ (see *Ruppert, 1988*, *Polyak and Juditsky, 1992*).

It is postprocessing:
Compute $\boldsymbol{\Theta}_n$ by usual RM procedure, but use $\overline{\boldsymbol{\Theta}}_n$ as an estimator of $\theta^\star$.

In your opinion, is averaging useful far from the minimizer?

Stochastic gradient descent - Abscissa averaging (II)

Back to the example on estimating a mean

Let $f(\theta) = \frac{1}{2}\,\mathrm{E}\left[(N-\theta)^2\right]$ for a r.v. N, say $N \sim \mathcal{N}(\mu, \sigma^2)$. It is minimal in

$$\theta^\star = \mathrm{E}\left[N\right].$$

Maximum Likelihood Estimator

Given an iid sequence $N_1, \ldots, N_n$ distributed as N, $\widehat{\theta^\star}_{MLE} = \frac{1}{n}(N_1 + \ldots + N_n)$, and

$$\mathrm{Var}\left[\widehat{\theta^\star}_{MLE}\right] = \frac{\sigma^2}{n}.$$

Robbins-Monro like with constant steps

$\Theta_{j+1} = \Theta_j + \mathsf{a}(N_j - \Theta_j)$, with here a **constant**. After n steps:

- Without averaging, one can bound the error (bias + variance): (please check!)

$$\mathrm{E}\left[(\Theta_{n+1} - \theta^\star)^2\right] \leq \mathrm{e}^{-2n\mathsf{a}}(\theta_1 - \theta^\star)^2 + \mathsf{a}\sigma^2$$

- With averaging over n values, $\overline{\Theta}_n := \frac{1}{n}\sum_{j=1}^{n}\Theta_n$, I get (please check!)

$$\mathrm{E}\left[(\overline{\Theta}_{n+1} - \theta^\star)^2\right] \leq \frac{1}{\mathsf{a}^2 n^2}(\theta_1 - \theta^\star)^2 + \frac{1}{n}\sigma^2$$

QUIZZ What if we set the constant step $\mathsf{a} = 1/n$, $\mathsf{a} = 1/\sqrt{n}$? compare with MLE.

Robbins Monro - Take home message

On Robbins-Monro SGD algorithm

1. Robbins-Monro for minimization, where $\mathrm{E}\left[\widehat{\mathbf{G}}(\theta)\right] = \nabla f(\theta)$:

$$\mathbf{\Theta}_{n+1} = \mathbf{\Theta}_n - \mathsf{a}_n \widehat{\mathbf{G}}(\mathbf{\Theta}_n)$$

2. noisy gradient must be known $\Longrightarrow$ used in Neural Networks.
3. Converges in a quite general setting, non necessarily convex
4. f bounded, not to steepy, not too flat

On step sizes = "learning rates"

1. Step sizes must decrease. Constant step leads to a noise ball
2. Not too small nor to high. $\sum \mathsf{a}_n = +\infty$, $\sum \mathsf{a}_n^2 < +\infty$:
   Too small steps: exhausted far from $\theta^\star$. Too high: bounces around $\theta^\star$.
3. Suitable steps $\mathsf{a}_n = \mathsf{a}/n$, $\mathsf{a} > 0$.
   Averaging allows to increase step sizes and fasten convergence.

to be continued...

in the rest of the lecture, we will detail more results on

- When the gradient is estimated (Kiefer-Wolfowitz)
- Acceleration and adaptation techniques
- Applications to neural networks and machine learning....

Stochastic Gradient Descent

Let us continue with finding the minimizer of a noisy function.

when the noisy gradient is unknown this is still:

Stochastic Gradient Descent (unknown gradient)

Stochastic gradient descent

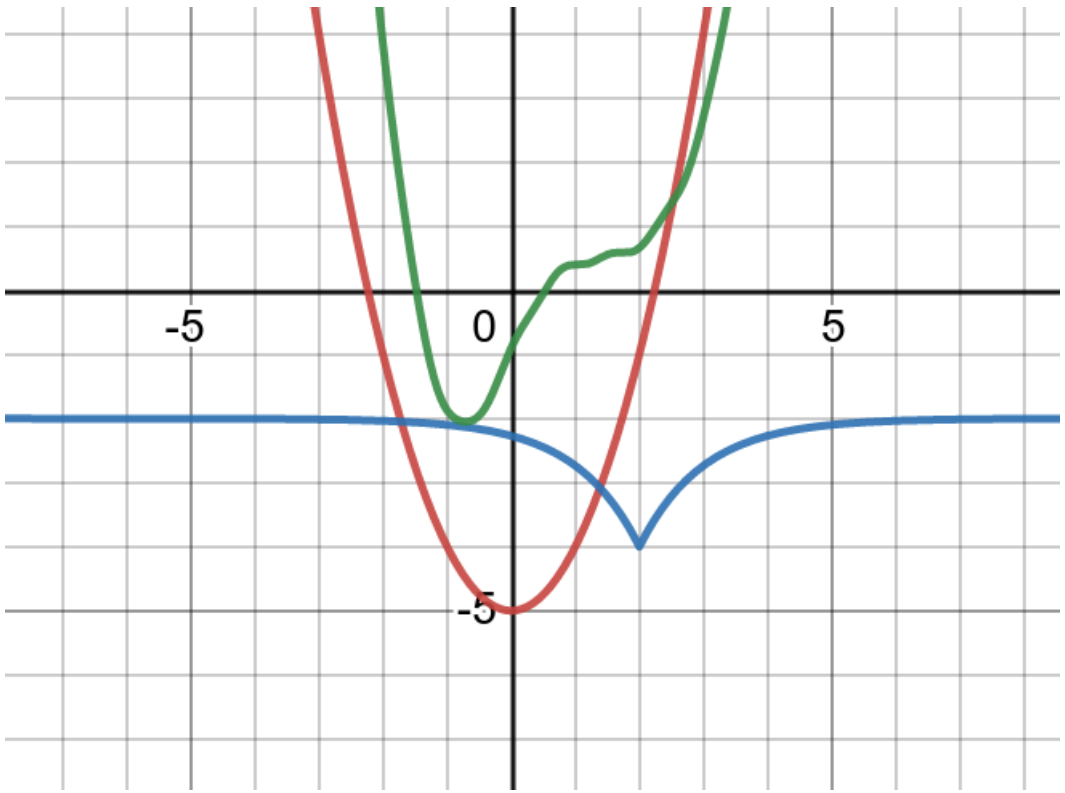

Jack Kiefer and Jacob Wolfowitz.

Stochastic gradient descent - 1D model

Assume that one observes real random variables $F(\theta)$, $\theta \in \mathbb{R}$.

- mutually independent
- such that $f(\theta) = \mathrm{E}[F(\theta)]$ exists
- $f(.)$ is decreasing to a minimum at abscissa $\theta^\star$, then increasing

...other conditions coming...



Stochastic gradient descent - 1D

Kiefer-Wolfowitz (SGD)

Stochastic gradient descent with estimated gradient *Kiefer and Wolfowitz*, 1952:

$$\Theta_{n+1} = \Theta_n - a_n \frac{F(\Theta_n + c_n) - F(\Theta_n - c_n)}{2c_n} \tag{10}$$

starting from $\Theta_1 = \theta_1 \in \mathbb{R}$, with given sequences $\{a_n\}$ and $\{c_n\}$.

- $\frac{F(\Theta_n+c_n)-F(\Theta_n-c_n)}{2c_n}$ estimated gradient (random!).
- a_n step size in the direction of the random gradient.

QUIZZ intuitively, which conditions on F ?

QUIZZ intuitively, which conditions on $\{a_n\}$, on $\{c_n\}$?

Stochastic gradient descent - Conditions

Conditions on $F(.)$ and $f(.)$ (without derivatives, see *Kiefer and Wolfowitz, 1952*)

- **C0**: finite variance of $F(.)$

  $\mathrm{Var}\left[F(\theta)\right] < \infty$ for all θ

- **C1**: Lipschitz condition near $\theta^\star$

  $\exists \beta, B: \ |\theta' - \theta^\star| + |\theta'' - \theta^\star| < \beta \implies |f(\theta') - f(\theta'')| < B|\theta' - \theta''|$

- **C2**: bounded variations on $f(.)$ Lipschitz ok but not only, e.g. f not continuous

  $\exists \rho, R: \ |\theta' - \theta''| < \rho \implies |f(\theta') - f(\theta'')| < R$

- **C3**: absolute derivative of $f(.)$ not too small far from $\theta^\star$

  $\forall \delta > 0, \exists \pi(\delta): \ |\theta - \theta^\star| > \delta \implies \inf_{0<\epsilon<\delta/2} \frac{|f(\theta+\epsilon) - f(\theta-\epsilon)|}{\epsilon} > \pi(\delta)$

Conditions on positive coefficients $\{a_n\}$ and $\{c_n\}$

- $c_n \to 0$, $a_n \to 0$ (implicit)
- **H1**: $\sum a_n = +\infty$

- **H2**: $\sum a_n c_n < +\infty$

- **H3**: $\sum a_n^2 c_n^{-2} < +\infty$

▸ See convergence proof

What have you learned?

- The way we study convergence: $\mathrm{E}\left[(\Theta_n - \theta^\star)^2\right]$ and $\mathrm{P}\left[|\Theta_n - \theta^\star| > \delta\right] \to 0$, using the recursion.
- The idea that there are conditions on $f(.)$:

  bounded, not too steepy, not to flat

- The idea that there are conditions on $\{a_n\}$, $\{c_n\}$

  both $\{a_n\}$, $\{c_n\} \downarrow 0$ not too fast, not too slow

- The idea of a bias/variance tradeoff for SGD.
- Keep in mind the situations where it may fail!

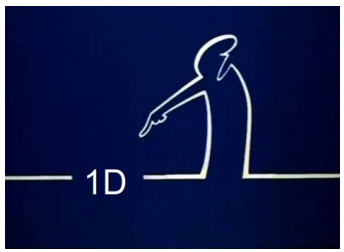
what happens if $\{a_n\} \downarrow 0$ too fast? too slowly?

what happens if $\{c_n\} \downarrow 0$ too fast? too slowly?

Keep home message: you can try numerically million things, but theory helps!

Stochastic gradient descent - Multidimensional

Multidimensional adaptation of Kiefer-Wolfowitz



- One observes real random variables $F(\theta)$, $\theta \in \mathbb{R}^d$, mutually indep.
- such that the unknown function $f(\theta) = \mathrm{E}[F(\theta)]$ exists.
- One look for the minimum $\theta^\star = \operatorname{argmin}_{\theta \in \mathbb{R}^d} f(\theta)$.

Kiefer-Wolfowitz-Blum (SGD)

Multidimensional Stochastic gradient descent with estimated gradient *Blum*, 1954:

$$\mathbf{\Theta}_{n+1} = \mathbf{\Theta}_n - a_n \widehat{\mathbf{G}}(\mathbf{\Theta}_n) \tag{11}$$

with given $\mathbf{\Theta}_1 = \theta_1$, $\widehat{\mathbf{G}}(\theta) = \left(\frac{F(\theta + c_n e_i) - F(\theta)}{c_n} \right)_{i=1,\dots,d}$ and $e_k = \left(\mathbb{1}_{\{i=k\}} \right)_{i=1,\dots,d}$.
Then under next slide's conditions on $\{a_n\}$, $\{c_n\}$ and $F(.)$,

$$\mathbf{\Theta}_n \overset{a.s.}{\to} \theta^\star.$$

- Notice the stronger convergence result obtained by *Blum*, 1954
- Here one sided gradient approx., but still $d + 1$ evaluations of $F(.)$ at each step

Stochastic gradient descent - Multidimensional conditions

Multivariate case, conditions on $F(.)$ and $f(.)$, *Blum, 1954*

- **C0'**: finite variance of $F(.)$

  $\mathrm{Var}\left[F(\theta)\right] < \infty$ for all $\theta \in \mathbb{R}^d$

- **C1'**: continuity of $f(.)$ and derivatives

  $f(.)$ is continuous, with continuous first and second derivatives

- **C2'** second partial derivatives of $f(.)$ are bounded

  $\partial f(\theta)/\partial\theta_i\partial\theta_j$ bounded for $i, j = 1, \ldots, d$, for all $\theta \in \mathbb{R}^d$

- **C3'**: far from $\theta^\star$, $f(.)$ far enough from optimum with gradient not too small.

  $\forall \delta > 0, \exists \pi(\delta) : \|\theta - \theta^\star\| > \delta \implies |f(\theta^\star) - f(\theta)| \geq \pi(\delta)$ & $\|\nabla f(\theta)\| \geq \pi(\delta)$

Conditions on positive coefficients $\{a_n\}$ and $\{c_n\}$: unchanged.

- $c_n \to 0$, $a_n \to 0$ (implicit)
- **H1**: $\sum a_n = +\infty$

- **H2**: $\sum a_n c_n < +\infty$

- **H3**: $\sum a_n^2 c_n^{-2} < +\infty$

Imagine what can happen if one condition on $F(.)$ and $f(.)$ is not reached.

Stochastic gradient descent - practical details

Estimation of the gradient

- $\widehat{\mathbf{G}}(\boldsymbol{\theta}) = \left(\frac{F(\boldsymbol{\theta}+c_n e_i) - F(\boldsymbol{\theta}-c_n e_i)}{2c_n} \right)_{i=1,\dots,d} \quad \Longrightarrow$ $2d$ new evaluations of $F(.)$
- $\widehat{\mathbf{G}}(\boldsymbol{\theta}) = \left(\frac{F(\boldsymbol{\theta}+c_n e_i) - F(\boldsymbol{\theta})}{c_n} \right)_{i=1,\dots,d} \quad \Longrightarrow$ $d+1$ new evaluations of $F(.)$:
- $\widehat{\mathbf{G}}(\boldsymbol{\theta}) = \left(\frac{F(\boldsymbol{\theta}+c_n \boldsymbol{\Delta}_n) - F(\boldsymbol{\theta}-c_n \boldsymbol{\Delta}_n)}{2c_n (\boldsymbol{\Delta}_n)_i} \right)_{i=1,\dots,d} \quad \Longrightarrow$ 2 new evaluations of $F(.)$

  $\boldsymbol{\Delta}_n$ are iid centered random vectors, with finite covariances and $\mathrm{E}\left[|(\boldsymbol{\Delta}_n)_i|^{-1} \right]$ bounded, e.g. Rademacher distribution. This is called Simultaneous perturbation stochastic approximation (SPSA), *Bhatnagar et al.*, 2013

Other practical details

- Suitable $\{a_n\}$, $\{c_n\}$ in KW: $a_n \sim a/n^\alpha$ and $c_n \sim c/n^\gamma$, $a > 0$, $c > 0$, with $\gamma \in (0,\, 1/2)$ and $\alpha \in (\max(\gamma + 1/2, 1 - 2\gamma),\, 1]$, cf. *Dippon*, 2003. Optimal $a_n = a/n$, $c_n = c/n^{1/4}$ or $c/n^{1/6}$ in *Broadie et al.*, 2011's Prop.1 or 4 setting.
- When $f(\theta^\star)$ is of importance, *Mokkadem and Pelletier*, 2007 allows to estimate jointly $\theta^\star$ and $f(\theta^\star)$, with convergence results and acceleration.

Kiefer Wolfowitz - Take home message

On Kiefer-Wolfowitz-Blum SGD algorithm

1. Kiefer-Wolfowitz-Blum for minimization, with $\widehat{\mathbf{G}}(\theta) = \left(\frac{F(\theta+c_n e_i)-F(\theta)}{c_n}\right)_{i=1,\dots,d}$.

$$\mathbf{\Theta}_{n+1} = \mathbf{\Theta}_n - a_n \widehat{\mathbf{G}}(\mathbf{\Theta}_n)$$

   suited when no knowledge on the gradient (otherwise use it!), other $\widehat{\mathbf{G}}$ possible.
2. Converges in a quite general setting, non necessarily convex
3. f bounded, not to steepy, not too flat

On step sizes = "learning rates"

1. Step sizes must decrease. Constant step leads to a noise ball
2. Not too small nor to high. $\sum a_n = +\infty$, $\sum a_n^2 < +\infty$:
   Too small steps: exhausted far from θ. Too high: bounces around θ.
3. Suitable steps $a_n = a/n$ and $c_n = c/n^\gamma$, $\gamma \in (0, 1/2)$.
   Averaging allows to increase step sizes and fasten convergence.

1. Motivation
   - Why?
   - How?
2. Stochastic Approximation: finding roots
   - Robbins-Monro for SA
3. Stochastic Gradient Descent
   - Robbins-Monro for SGD
   - Kiefer-Wolfowitz for SGD
   - Acceleration, adapted gradient descent
4. Application to Machine Learning
   - Applications
   - Conclusion

Stochastic gradient descent - Acceleration (I)

Acceleration techniques (here presented for variable steps)

Note: acceleration techniques can change optimal step sizes $\{a_n\}$ and $\{c_n\}$!

- **Abscissa averaging** (seen before) *Ruppert*, 1988, *Polyak and Juditsky*, 1992.

$$\begin{cases} \boldsymbol{\Theta}_{n+1} &= \boldsymbol{\Theta}_n - a_n \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n) \\ \overline{\boldsymbol{\Theta}}_{n+1} &= \frac{1}{n_0} \sum_{j=n-n_0}^{n+1} \boldsymbol{\Theta}_j \end{cases} \tag{12}$$

  for a given lag n_0, $1 \leq n_0 \leq n$. Θ_n computed by usual procedure, but we use $\overline{\Theta}_n$ as an estimator of $\theta^\star$ instead of $\boldsymbol{\Theta}_n$. This is postprocessing.

- **Classical Momentum** (Gradient averaging) *Polyak*, 1964:

$$\begin{cases} \mathbf{M}_{n+1} = \beta_n \mathbf{M}_n - a_n \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n) \\ \boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n + \mathbf{M}_{n+1} \end{cases} \tag{13}$$

  $\beta_n \in [0, 1]$ is the momentum coefficient. Persistence in the direction $\mathbf{M}_n$, called velocity, to smooth direction and reduce bouncing.

- **Nesterov Momentum** (*Nesterov's Accelerated Gradient, NAG*) *Nesterov*, 2003

$$\begin{cases} \mathbf{M}_{n+1} = \beta_n \mathbf{M}_n - a_n \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n + \beta_n \mathbf{M}_n) \\ \boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n + \mathbf{M}_{n+1} \end{cases} \tag{14}$$

  Proven faster convergence in some settings, put more faith in the future gradient.

Stochastic gradient descent - Acceleration (II)

Interlude: accelerations are based on averaging!

We will see averaging for abscissas, gradients, squared gradient components...

Consider real sequences $\{u_n\}$ and $\{v_n\}$ with $\beta \in [0, 1[$ and recurrence

$$u_{n+1} = \beta u_n + (1 - \beta) v_n \tag{15}$$

Then

$$\left\{ \begin{array}{lcl} u_1 & = & \beta\, u_0 + (1-\beta)\, v_0 \\ u_2 & = & \beta^2\, u_0 + \beta(1-\beta)\, v_0 + (1-\beta)\, v_1 \\ u_3 & = & \beta^3\, u_0 + \beta^2(1-\beta)\, v_0 + \beta(1-\beta)\, v_1 + (1-\beta)\, v_2 \\ & \dots & \end{array} \right. \tag{16}$$

Consequence if $u_0 = 0$

- u_n is a weighted sum of $v_0, \dots, v_{n-1}$
- v_n has more weights than v_{n-1}, etc.: $\beta^2(1-\beta) < \beta(1-\beta) < (1-\beta)$
- sum of weights is $1 - \beta^n$ (excluding the weight of $u_0 = 0$)
- $u_n/(1-\beta^n)$ is a moving average of $\{v_i\}_{i<n}$ named "*exponential moving average*"

Stochastic gradient descent - Acceleration (III)

Adapted gradient descent illustration

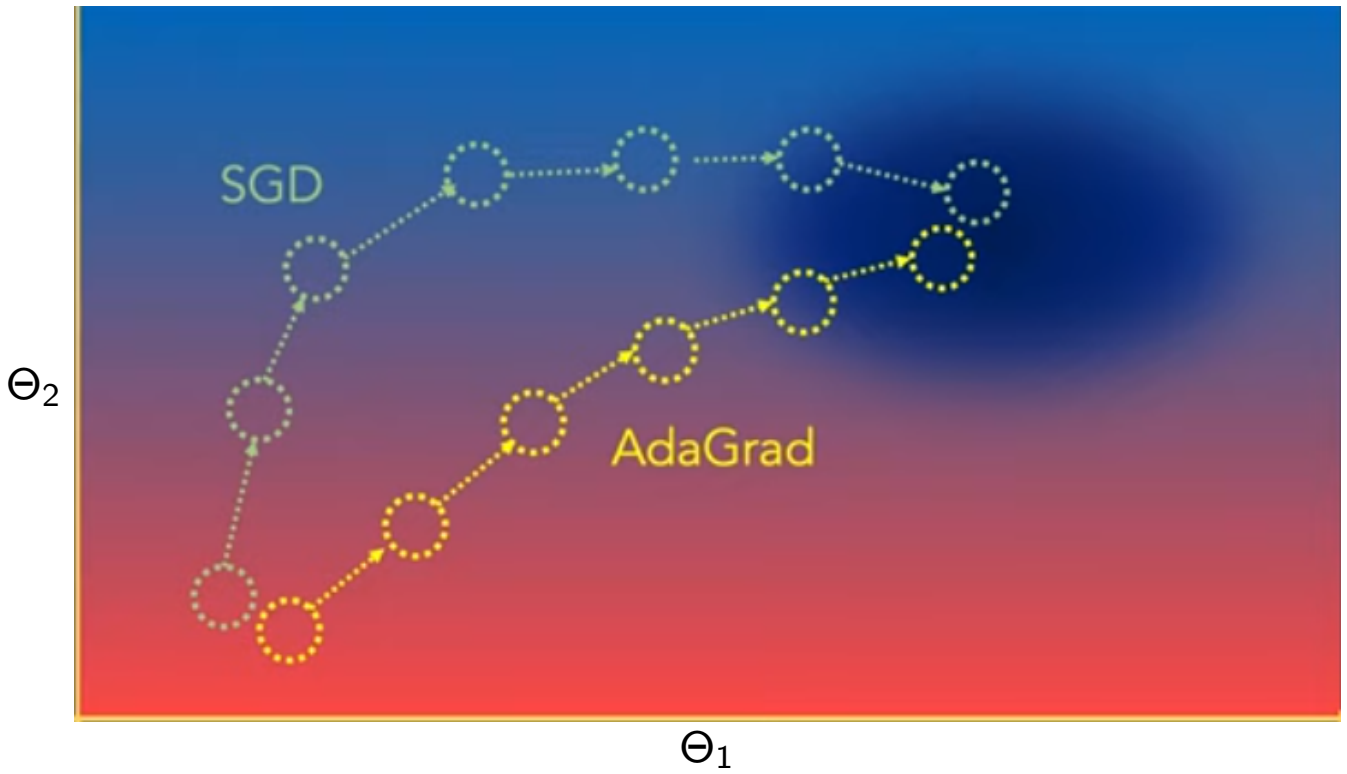


Illustration from `https://youtu.be/NE88eqLngkg`

Stochastic gradient descent - Acceleration (IV)

Adapted gradient descent : adapt step(s) size along iterations

See for example explanations in https://youtu.be/NE88eqLngkg

- **AdaGrad** (*adaptative gradient*) *Duchi et al.*, 2011:

! componentwise operations !

$$\boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \frac{a}{10^{-8} + \sqrt{\sum_{j=1}^{n} \widehat{\mathbf{G}}(\boldsymbol{\Theta}_j)^2}} \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n) \tag{17}$$

reduce steps along dimensions where gradients have moved a lot.

- **RMSprop** (*root mean square propagation*) *Tieleman and Hinton*, 2012:

! componentwise operations !

$$\begin{cases} \mathbf{V}_{n+1} = \beta \mathbf{V}_n + (1-\beta)\widehat{\mathbf{G}}(\boldsymbol{\Theta}_n)^2 \\ \boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \frac{a}{10^{-8} + \sqrt{\mathbf{V}_{n+1}}} \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n) \end{cases} \tag{18}$$

Same idea with decay factor β allowing step rates to re-increase.

Resulting steps in most simple cases (parabola...)?

Stochastic gradient descent - Acceleration (III)

Adapted gradient descent illustration

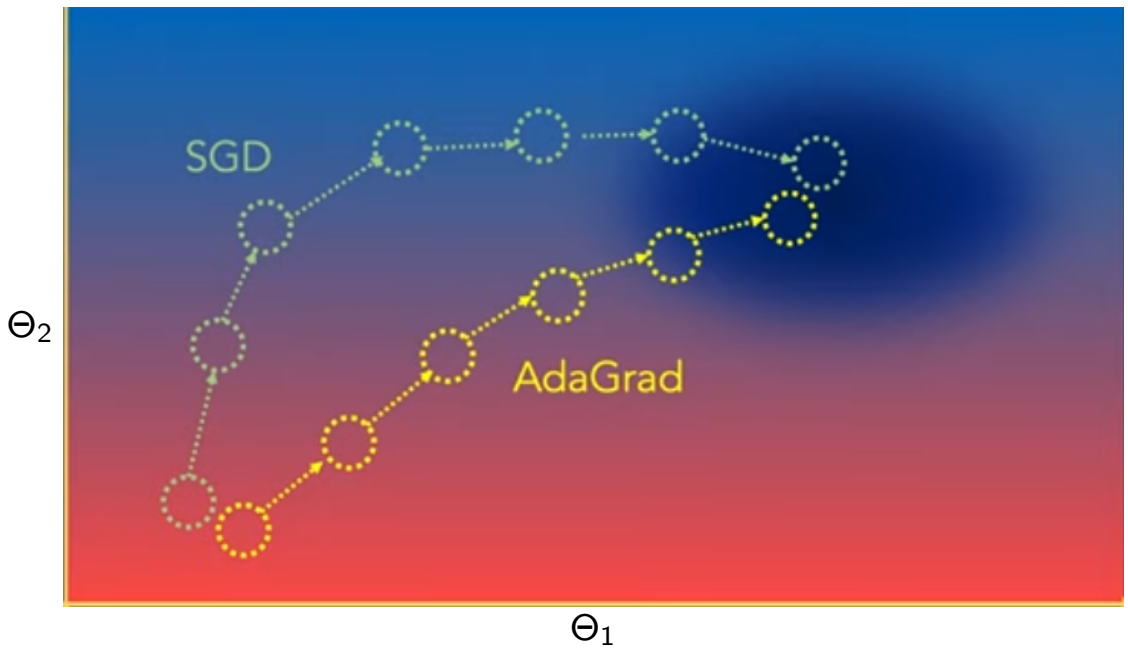


Illustration from `https://youtu.be/NE88eqLngkg`, see explanations in this video.

As gradient second components are larger, steps along this dimension are reduced.

Stochastic gradient descent - Acceleration (V)

- **ADAM** (*Adaptive Moment Estimation*) *Kingma and Ba*, 2014, https://arxiv.org/abs/1412.6980

! componentwise operations !

$$\Theta_{n+1} = \Theta_n - \frac{a}{10^{-8} + \sqrt{\hat{\mathbf{V}}_{n+1}}} \hat{\mathbf{M}}_{n+1} \tag{19}$$

$$\text{with} \begin{cases} \mathbf{M}_{n+1} = \beta_1 \mathbf{M}_n + (1-\beta_1)\widehat{\mathbf{G}}(\Theta_n) & \text{(momentum persistance, direction)} \\ \mathbf{V}_{n+1} = \beta_2 \mathbf{V}_n + (1-\beta_2)\widehat{\mathbf{G}}(\Theta_n)^2 & \text{(for adapting amplitude by dimension)} \end{cases}$$

$$\text{and final rescaling} \begin{cases} \hat{\mathbf{M}}_{n+1} = \mathbf{M}_{n+1}/(1-\beta_1^{n+1}) \\ \hat{\mathbf{V}}_{n+1} = \mathbf{V}_{n+1}/(1-\beta_2^{n+1}) \end{cases} \tag{20}$$

See for example explanations in https://youtu.be/NE88eqLngkg

1. Increases (resp.reduce) steps along dimensions where gradients are small (large)
2. Gradient oscillations induce step reduction, since $\hat{\mathbf{M}}$ is averaged
3. Invariance by gradient components rescaling
4. Bounded effective steps $\|\Theta_{n+1} - \Theta_n\|$: hint on parameter a
5. Not so old (last decade), heavily cited (230K!), active research field.

intuitively, do steps sizes satisfy Robbins Monro assumptions? try basic cases?

Stochastic gradient descent - Acceleration (VI)

ADAM in practice

"*Good default settings for the tested machine learning problems are* $a = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$."

Kingma and Ba, 2014

ADAM guarantees ☺

Define a *regret* (from independent realizations of the noisy function F):

$$\mathcal{R}_n = \sum_{i=1}^{n} \left(F(\boldsymbol{\Theta}_i) - F(\theta^\star) \right) , \quad \text{necessarily} \geq 0 \tag{21}$$

With decaying steps $a_n = \frac{a}{\sqrt{n}}$, and decaying rates $\beta_{1,n}$, under assumptions of their Corollary 4.2, *Kingma and Ba*, 2014 prove that the average regret converges to 0:

$$\frac{\mathcal{R}_n}{n} = \mathcal{O}\left(n^{-1/2}\right) \quad \text{as } n \to \infty . \tag{22}$$

With RM-like conditions $\sum a_n = +\infty$, $\sum a_n^2 < \infty$, other recent a.s. CV results : *Barakat and Bianchi*, 2021, Th 5.2 (novel decreasing stepsize version of Adam)

Accelerations and Improvements - Take home message

On abcissa averaging

1. Postprocessing
2. Allows larger step sizes
3. Not useful far from optimum

On momentum and adapted gradient

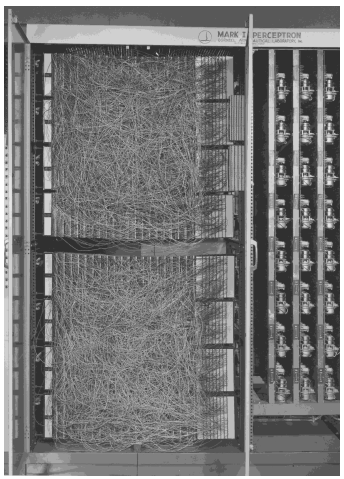
1. Momentum. Acts on the direction: persistence effect (like inertia or mass).
2. Adapted gradient. Acts on step size: reduce steps along some dimensions.
3. Combination. ADAM rough componentwise general expression:

$$\Theta_{n+1} = \Theta_n - \frac{a_n}{\sqrt{\hat{\mathbf{V}}_{n+1}}} \hat{\mathbf{M}}_{n+1} \tag{23}$$

where $\hat{\mathbf{M}}_{n+1}$ and $\hat{\mathbf{V}}_{n+1}$ are moving averages of previous $\widehat{\mathbf{G}}(.)$ and $\widehat{\mathbf{G}}(.)^2$.

4. Application to Machine Learning

Application to Machine Learning



Mark I Perceptron (1958) and El Capitan (2025, 30MW)

Robbins-Monro 1951 $<$ Kiefer-Wolfowitz 1952 $<$ Blum 1954 $<$ Perceptron 1958

Preliminary : why picking data makes it noisy?

Loss on the whole dataset

x	y
10	100
20	200
30	300
40	400
50	500
60	600

One tries to fit, for an unknown parameter $\theta^\star$,

$$y_i = h(x_i, \theta^\star)$$

A deterministic mean loss on whole data, for candidate parameter θ:

$$\ell(\theta) = \frac{1}{6}\sum_{i=1}^{6} (h(x_i, \theta) - y_i)^2$$

Loss on a single random row

Pick a random row number $D \in \{1, ..., 6\}$, this is a dice! The loss on this row is

$$\ell_D(\theta) = (h(x_D, \theta) - y_D)^2 = (h(10D, \theta) - 100D)^2 = \text{Random Variable!}$$

$$\text{observe that} \quad \mathrm{E}\left[\ell_D(\theta)\right] = \sum_{i=1}^{6} \underbrace{\mathrm{P}\left[D = i\right]}_{1/6} (h(x_i, \theta) - y_i)^2 = \ell(\theta)$$

$$\text{so that} \quad \ell_D(\theta) = \ell(\theta) + \mathit{Noise}(\theta)$$

☺ $\nabla \ell_D(\theta)$ is a known noisy gradient: $\mathrm{E}\left[\nabla \ell_D(\theta)\right] = \sum_{i=1}^{6} \frac{1}{6} \nabla \left(h(x_i, \theta) - y_i\right)^2 = \nabla \ell(\theta)$

Machine Learning classical presentation

For inputs $\mathbf{x}_i$, predictions $h(\mathbf{x}_i, \theta)$ and true outputs y_i, $i = 1, ..., n_{\text{obs}}$.

Function to minimize in θ, $f(\theta) = \ell(\theta)$

- "***full batch***": One measures the error of the fit by a loss function (n_{obs} often huge!)

$$\ell(\theta) = \frac{1}{n_{\text{obs}}} \sum_{i=1}^{n_{\text{obs}}} (h(\mathbf{x}_i, \theta) - y_i)^2$$

- "***sgd***": random loss on a random data row K

$$\ell_K(\theta) = (h(\mathbf{x}_K, \theta) - y_K)^2$$

- "***minibatch***": random loss on a random subset of rows $\mathcal{K}$:

$$\ell_{\mathcal{K}}(\theta) = \frac{1}{\operatorname{card} \mathcal{K}} \sum_{K \in \mathcal{K}} (h(\mathbf{x}_K, \theta) - y_K)^2$$

Known Noisy Gradients $\widehat{\mathbf{G}}_{\mathcal{K}}(\theta = \nabla \ell_{\mathcal{K}}(\theta)$

Under fair sampling conditions, one can check that

$$\mathrm{E}\left[\ell_K(\theta)\right] = \mathrm{E}\left[\ell_{\mathcal{K}}(\theta)\right] = \ell(\theta) \quad \text{and thus} \quad \mathrm{E}\left[\nabla \ell_K(\theta)\right] = \mathrm{E}\left[\nabla \ell_{\mathcal{K}}(\theta)\right] = \nabla \ell(\theta)$$

SGD Algorithm for Machine Learning

SGD algorithm in the ML case, case $\operatorname{card} \mathcal{K} = 1$

Input: learning rate(s) a or $\{\mathsf{a}_n\}$
Input: initial parameter Θ_1

Randomly shuffle samples in the training set.
$n \leftarrow 1$
repeat
  $\Theta_{n+1} \leftarrow \Theta_n - \mathsf{a}_n \nabla \ell_n(\Theta_n)$ ▷ $\nabla \ell_n$ gradient loss for the n^{th} (shuffled) data row
  $n \leftarrow n + 1$
until stopping criterion

Output: Θ_n

Other details in Appendix

More examples in the Appendix:

- A general Loss Function example:
  - ▸ Loss Function example
- A Toy example (Dice):
  - ▸ Toy example
- A Regression example:
  - ▸ Regression example

Specific case of Neural Networks

- A general AI example:
  - ▸ AI example
- A Neural Net example:
  - ▸ Neural Net example
- Calculation of the gradient for Neural Networks (backpropagation):
  - ▸ Details on gradient backpropagation

Application to Machine Learning

General expression for gradients

Given a data $\{\mathbf{row}_i\}_{i=1,2,\dots}$, e.g. $\mathbf{row}_i = (\mathbf{x}_i, y_i)$
Given a (random) sample $\mathcal{K}$ of row indices, e.g. $\mathcal{K} = \{2, 5, 8\}$,
The average loss to minimize in $\boldsymbol{\theta}$ is

$$f_{\mathcal{K}}(\boldsymbol{\theta}) = \mathrm{E}\left[\frac{1}{\operatorname{card}\mathcal{K}} \sum_{K \in \mathcal{K}} \ell(\boldsymbol{\theta}, \mathbf{row}_K)\right]$$

Assuming $f(\boldsymbol{\theta}) := f_{\mathcal{K}}(\boldsymbol{\theta})$ does not depend on $\mathcal{K}$, we get by linearity

$$\nabla f(\boldsymbol{\theta}) := \mathrm{E}\left[\frac{1}{\operatorname{card}\mathcal{K}} \sum_{K \in \mathcal{K}} \nabla \ell(\boldsymbol{\theta}, \mathbf{row}_K)\right] =: \mathrm{E}\left[\widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\theta})\right]$$

$\widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\theta})$ is an estimator of $\nabla f(\boldsymbol{\theta})$, easy to compute if $\nabla \ell(\boldsymbol{\theta}, .)$ is known.

Noisy gradient assumption

One can sample $\mathcal{K}$ and compute a noisy gradient $\widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\theta})$ such that

$$\mathrm{E}\left[\widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\theta})\right] = \nabla f(\boldsymbol{\theta})$$

Application to Machine Learning - Batches (I)

Stochastic gradient descent, when $\nabla \ell$ is known (w.r.t. θ)

SGD for Machine Learning: when $\nabla \ell$ is known

Let the unbiased estimator of $\nabla f(\theta)$ given a (random) sample $\mathcal{K}$ be

$$\widehat{\mathbf{G}}_{\mathcal{K}}(\theta) := \frac{1}{\operatorname{card} \mathcal{K}} \sum_{K \in \mathcal{K}} \nabla \ell(\theta, \mathbf{row}_K) \tag{24}$$

Starting from a point $\mathbf{\Theta}_1 = \theta_1$, the SGD writes:

$$\mathbf{\Theta}_{n+1} = \mathbf{\Theta}_n - \mathsf{a}_n \widehat{\mathbf{G}}_{\mathcal{K}}(\mathbf{\Theta}_n) \tag{25}$$

Machine learning terminology (all can be seen as classical SGD on different functions)

- $\operatorname{card} \mathcal{K} = 1 \quad \Longrightarrow$ "***Stochastic Gradient Descent***"
- $1 < \operatorname{card} \mathcal{K} < \operatorname{card} \mathrm{Data} \quad \Longrightarrow$ "***Minibatch***"
- $\operatorname{card} \mathcal{K} = \operatorname{card} \mathrm{Data} \quad \Longrightarrow$ "**Full Batch**", "*Large-scale optimization*"

Step sizes a_n are usually called the "*learning rates*".

Application to Machine Learning - Batches (II)

Trade-off on the batch sizes

Since the gradient computation is $\mathcal{O}(\operatorname{card} \mathcal{K})$:

- Small batch size: heavy noise on gradient, fast gradient computation
- Large batch size: low noise on gradient, slow gradient computation

Practical considerations

- At the beginning of the descent, high noise is sufficient to get a direction.
- At the end, lower noise required.

Open question: for a given evaluation budget of F, is it better:

- to do more steps of the stochastic gradient descent?

- or to reduce the noise on the gradient?

Application to Machine Learning - Batches (III)

Batches - two steps intuition

Batch corresponds to averaging over outputs (not abscissa as previous averaging!).

Compare the following:

- Stochastic Gradient Descent with half constant steps $a/2$. After two steps :
  say, two iterations with step $a/2$, batches having size 1, cpu time=2

$$\Theta_{n+2} = \underbrace{\Theta_n - \frac{a}{2}\widehat{G}(\Theta_n)}_{\Theta_{n+1}} - \frac{a}{2}\widehat{G}\left(\underbrace{\Theta_n - \frac{a}{2}\widehat{G}(\Theta_n)}_{\Theta_{n+1}}\right) \tag{26}$$

- Minibatch with full step a, output averaging of two independent draws of $\widehat{G}()$:
  say, one iteration with step a, batch having size 2, cpu time=2

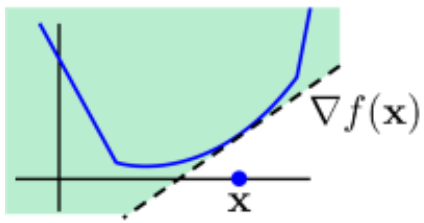
$$\Theta_{next} = \underbrace{\Theta_n - \frac{a}{2}\widehat{G}(\Theta_n) - \frac{a}{2}\widehat{G}\left(\Theta_n - 0^+\right)}_{\text{larger batch = averaging } \widehat{G}} \tag{27}$$

QUIZZ What do you understand from this, when steps are small?

QUIZZ During iterations, is it better to increase the batch size or to reduce steps?

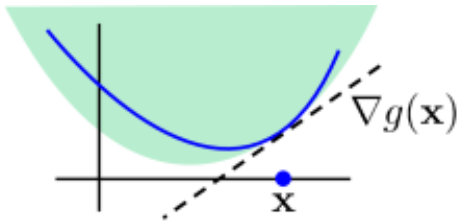
Convergence rates (0)

Notion of strongly convexity and smoothness



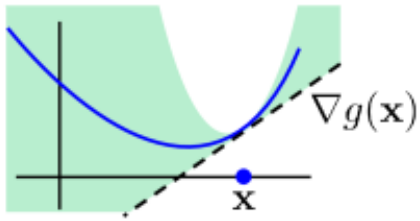
$f : \mathbb{R}^d \to \mathbb{R}$

CONVEX FUNCTION



$g : \mathbb{R}^d \to \mathbb{R}$

STRONGLY CONVEX FUNCTION



$g : \mathbb{R}^d \to \mathbb{R}$

STRONGLY SMOOTH CONVEX FUNCTION

images from www.arxiv-vanity.com/papers/1712.07897/, sec. 2.1, see *Jain and Kar*, 2017

- If f convex:

$$f\left(\theta'\right) \geq f\left(\theta\right) + \left\langle \theta' - \theta\,,\, \nabla f\left(\theta\right) \right\rangle$$

- If f μ-strongly convex and L-strongly smooth:

$$f\left(\theta'\right) \quad \geq \quad f\left(\theta\right) + \left\langle \theta' - \theta\,,\, \nabla f\left(\theta\right) \right\rangle + \frac{\mu}{2}\|\theta' - \theta\|^2$$

$$f\left(\theta'\right) \quad \leq \quad f\left(\theta\right) + \left\langle \theta' - \theta\,,\, \nabla f\left(\theta\right) \right\rangle + \frac{L}{2}\|\theta' - \theta\|^2$$

QUIZZ Are these assumptions reasonable for neural networks?

Convergence rates (I)

Deterministic case (without twice differentiability assumption ☺)

Deterministic case: linear convergence rate

Assume f μ-strongly convex and L-strongly smooth. Then

$$\|\Theta_{n+1} - \theta^\star\|^2 \leq \|\Theta_n - \theta^\star\|^2 |1 - a_n\mu| \,, \quad \forall a_n < 1/L \,.$$

Proof: gradient descent is $\Theta_{n+1} - \theta^\star = \Theta_n - \theta^\star - a_n \nabla f(\Theta_n)$, hence

$$\|\Theta_{n+1} - \theta^\star\|^2 = \|\Theta_n - \theta^\star\|^2 - 2a_n \langle \Theta_n - \theta^\star \,, \nabla f(\Theta_n) \rangle + a_n^2 \|\nabla f(\Theta_n)\|^2$$

- from μ-strong convexity: [hint, write $f(\theta^\star) \geq f(\Theta_n) + \dots$]

$$\langle \Theta_n - \theta^\star \,, \nabla f(\Theta_n) \rangle \geq f(\Theta_n) - f(\theta^\star) + \frac{\mu}{2} \|\Theta_n - \theta^\star\|^2$$

- from L-strong smoothness: [hint, write $f(\Theta_n - \frac{1}{L}\nabla f(\Theta_n)) \leq f(\Theta_n) + \dots$]

$$\|\nabla f(\Theta_n)\|^2 \leq 2L \left(f(\Theta_n) - f(\theta^\star)\right)$$

- Combining the two previous bounds:

$$\|\Theta_{n+1} - \theta^\star\|^2 \leq \|\Theta_n - \theta^\star\|^2 (1 - a_n\mu) + 2a_n \left(f(\Theta_n) - f(\theta^\star)\right)(a_n L - 1)$$

Convergence rates (II)

Stochastic case (without twice differentiability assumption ☺)

Stochastic case

Assume as previously strong μ-convexity.

Assume $\exists B$ such that $\mathrm{E}\left[\|\widehat{\mathbf{G}}(\theta)\|^2\right] \leq B^2$ for all θ.

$$\text{then} \quad \mathrm{E}\left[\|\boldsymbol{\Theta}_{n+1} - \theta^\star\|^2\right] \leq \mathrm{E}\left[\|\boldsymbol{\Theta}_n - \theta^\star\|^2\right] |1 - \mathsf{a}_n \mu| + \mathsf{a}_n^2 B^2 \,.$$

Proof: SGD is $\boldsymbol{\Theta}_{n+1} - \theta^\star = \boldsymbol{\Theta}_n - \theta^\star - \mathsf{a}_n \widehat{\mathbf{G}}(\boldsymbol{\Theta}_n)$, hence given $\boldsymbol{\Theta}_n = \theta_n$ (implicit cond.!)

$$\mathrm{E}\left[\|\boldsymbol{\Theta}_{n+1} - \theta^\star\|^2\right] = \|\theta_n - \theta^\star\|^2 - 2\mathsf{a}_n \left\langle \theta_n - \theta^\star \,,\, \mathrm{E}\left[\widehat{\mathbf{G}}(\theta_n)\right] \right\rangle + \mathsf{a}_n^2 \,\mathrm{E}\left[\|\widehat{\mathbf{G}}(\theta_n)\|^2\right]$$

- from μ-strong convexity, as $\mathrm{E}\left[\widehat{\mathbf{G}}(\theta_n)\right] = \nabla f(\theta_n)$,

$$\langle \theta_n - \theta^\star \,,\, \nabla f(\theta_n) \rangle \geq f(\theta_n) - f(\theta^\star) + \frac{\mu}{2} \|\theta_n - \theta^\star\|^2$$

- Combining this with B-regularity:

$$\mathrm{E}\left[\|\boldsymbol{\Theta}_{n+1} - \theta^\star\|^2 | \boldsymbol{\Theta}_n = \theta_n\right] \leq \|\theta_n - \theta^\star\|^2 |1 - \mathsf{a}_n \mu| + \mathsf{a}_n^2 B^2 \,.$$

hence the result when taking expectation over $\boldsymbol{\Theta}_n$.

Convergence rates (III)

Comparison

Convergence rates

Under previous assumptions strong μ-convexity, L-smoothness, and B-regularity. Assume $\forall n, \mathsf{a}_n = \mathsf{a}$ for simplicity, with $\mathsf{a} < 1/L$ and say, $\mu \ll 1$. Given $\boldsymbol{\Theta}_1 = \boldsymbol{\theta}_1$,

- Deterministic case: approx. large "full batch".

$$\|\boldsymbol{\Theta}_{n+1} - \boldsymbol{\theta}^\star\|^2 \leq \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^\star\|^2 (1 - \mathsf{a}\mu)^n \quad = \mathcal{O}\left(\mathrm{e}^{-\frac{\mu}{L} n}\right) \text{ if } \mathsf{a} \to 1/L\,.$$

- Stochastic case: "stochastic gradient descent"

$$\mathrm{E}\left[\|\boldsymbol{\Theta}_{n+1} - \boldsymbol{\theta}^\star\|^2\right] \leq \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^\star\|^2 (1 - \mathsf{a}\mu)^n + \underbrace{\mathsf{a}B^2/\mu}_{\text{noise ball}}\,.$$

if $\mathsf{a} = \mu/(1 + \mu^2 n) = \mathcal{O}\left(n^{-1}\right) \implies (1 - \mathsf{a}\mu)^n \sim \mathsf{a}/\mu$ and
$\mathrm{E}\left[\|\boldsymbol{\Theta}_{n+1} - \boldsymbol{\theta}^\star\|^2\right] = \mathcal{O}\left(n^{-1}\right)$

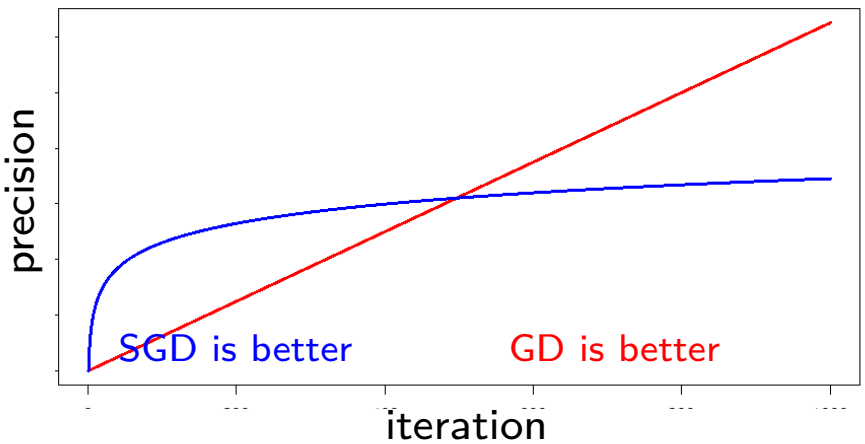
+details: www.college-de-france.fr/site/stephane-mallat/course-2019-03-20-11h15.htm

Can you see the "noise ball" of SGD when step sizes are constant ?

Convergence rates (IV)

"full batch": is it better to spend more time to get a deterministic gradient?

Assume it takes n_{obs} gradient evaluations to get a deterministic value  mmmh...



Precision of GD vs SGD as a function of iterations, the higher the better.

With a budget of n evaluations, compare :

- Deterministic GD: $\mathcal{O}\left(\mathrm{e}^{-\frac{\mu}{L}\frac{n}{n_{\text{obs}}}}\right)$, precision $-\ln \mathrm{e}^{-\frac{\mu}{L}\frac{n}{n_{\text{obs}}}} = \frac{\mu}{L}\frac{n}{n_{\text{obs}}}$ in red
- Stochastic GD: $\mathcal{O}\left(n^{-1}\right)$, precision $-\ln n^{-1} = \ln n$ in blue

$\Longrightarrow$ far from the optimum, the relative error of the sto. gradient can be neglected.

Application to Machine Learning - Take home message

On the SGD algorithm for Machine Learning

1. typical noisy loss $F_{\mathcal{K}}(\boldsymbol{\theta}) = \frac{1}{\operatorname{card}\mathcal{K}} \sum_{K\in\mathcal{K}} \ell(\boldsymbol{\theta}, \mathbf{row}_K)$ on a random sample $\mathcal{K}$.
2. if $\nabla\ell(.)$ is known, use Robbins-Monro with (noisy) gradients $\widehat{\mathbf{G}}_{\mathcal{K}}$:

$$\boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \mathsf{a}_n \widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\Theta}_n) \quad \text{with} \quad \widehat{\mathbf{G}}_{\mathcal{K}}(\boldsymbol{\theta}) := \frac{1}{\operatorname{card}\mathcal{K}} \sum_{K\in\mathcal{K}} \nabla\ell(\boldsymbol{\theta}, \mathbf{row}_K)$$

3. increasing minibatch size is close to decreasing learning rates, anyway, the precision must increase close to the optimum!

On step sizes = "learning rates"

1. Convergence far slower with noise. Constant step leads to a noise ball.
2. Not too small nor to high. $\sum \mathsf{a}_n = +\infty$, $\sum \mathsf{a}_n^2 < +\infty$: Too small steps: exhausted far from $\boldsymbol{\theta}^\star$. Too high: bounces around $\boldsymbol{\theta}^\star$.
3. Suitable steps $\mathsf{a}_n = \mathsf{a}/n$ leads to convergence in $\mathcal{O}\left(n^{-1}\right)$. With ideal strong convexity and smoothness that do not hold for neural networks!

Stochastic gradient descent and applications

Main interest

- very simple and easy to apply
- look for a local optimum, quite general convergence conditions
- well suited with information on the gradient, even with noise
- well suited in high dimension: gradient gives the right direction in each dimension
- proofs extendable to non differentiable functions (*relu*...)

Limits of the Stochastic Gradient Descent

- local optimization
- relies on many parameters, not always easy to tune
- choice of optimal step or batch sizes is not so easy
- different optimal parameters, depending on assumptions on the function
- few proofs in strongly convex case, but highly non-convex situations (neural net.)
- basically use only the previous information, Markovian behavior

... but huge literature and advances on all these points.

Application to Machine Learning

Practical manipulations

- Lecture 1 Elements on non-stochastic optimisation
  $\Longrightarrow$ see TP_Gradient notebook.

- Lecture 2 Neural network optimization, deterministic backpropagation
  $\Longrightarrow$ see TP_RNN notebook.

- Lecture 3 Elementary noisy function, RM, Averaging, KW intro
  $\Longrightarrow$ see TP_SGD notebook.

- Lecture 4 One neuron Optimization, extensions
  $\Longrightarrow$ see TP_SGDOneNeuron notebook.

  or

  Bonus TP, elicitability, expectiles
  $\Longrightarrow$ see TP_Elicitability notebook.

Some references I

[1] Barakat, A. and Bianchi, P. (2021). Convergence and dynamical behavior of the adam algorithm for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 31(1):244–274.

[2] Bhatnagar, S., Prasad, H., and Prashanth, L. (2013). *Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods*. Springer.

[3] Blum, J. R. (1954). Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics*, pages 737–744.

[4] Broadie, M., Cicek, D., and Zeevi, A. (2011). General bounds and finite-time improvement for the kiefer-wolfowitz stochastic approximation algorithm. *Operations Research*, 59(5):1211–1224.

[5] Cauchy, A. et al. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

[6] Dippon, J. (2003). Accelerated randomized stochastic optimization. *The Annals of Statistics*, 31(4):1260 – 1281.

[7] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Some references II

[8] Fouskakis, D. and Draper, D. (2002). Stochastic optimization: a review. *International Statistical Review*, 70(3):315–349.

[9] Jain, P. and Kar, P. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363.

[10] Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.

[11] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. `https://arxiv.org/abs/1412.6980`.

[12] Mokkadem, A. and Pelletier, M. (2007). A companion for the kiefer–wolfowitz–blum stochastic approximation algorithm. *The Annals of Statistics*, 35(4):1749–1772.

[13] Nesterov, Y. (2003). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

[14] Petrova, S. S. and Solov'ev, A. D. (1997). The origin of the method of steepest descent. *Historia Mathematica*, 24(4):361–375.

[15] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

Some references III

[16] Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855.

[17] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[18] Ruppert, D. (1988). Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering.

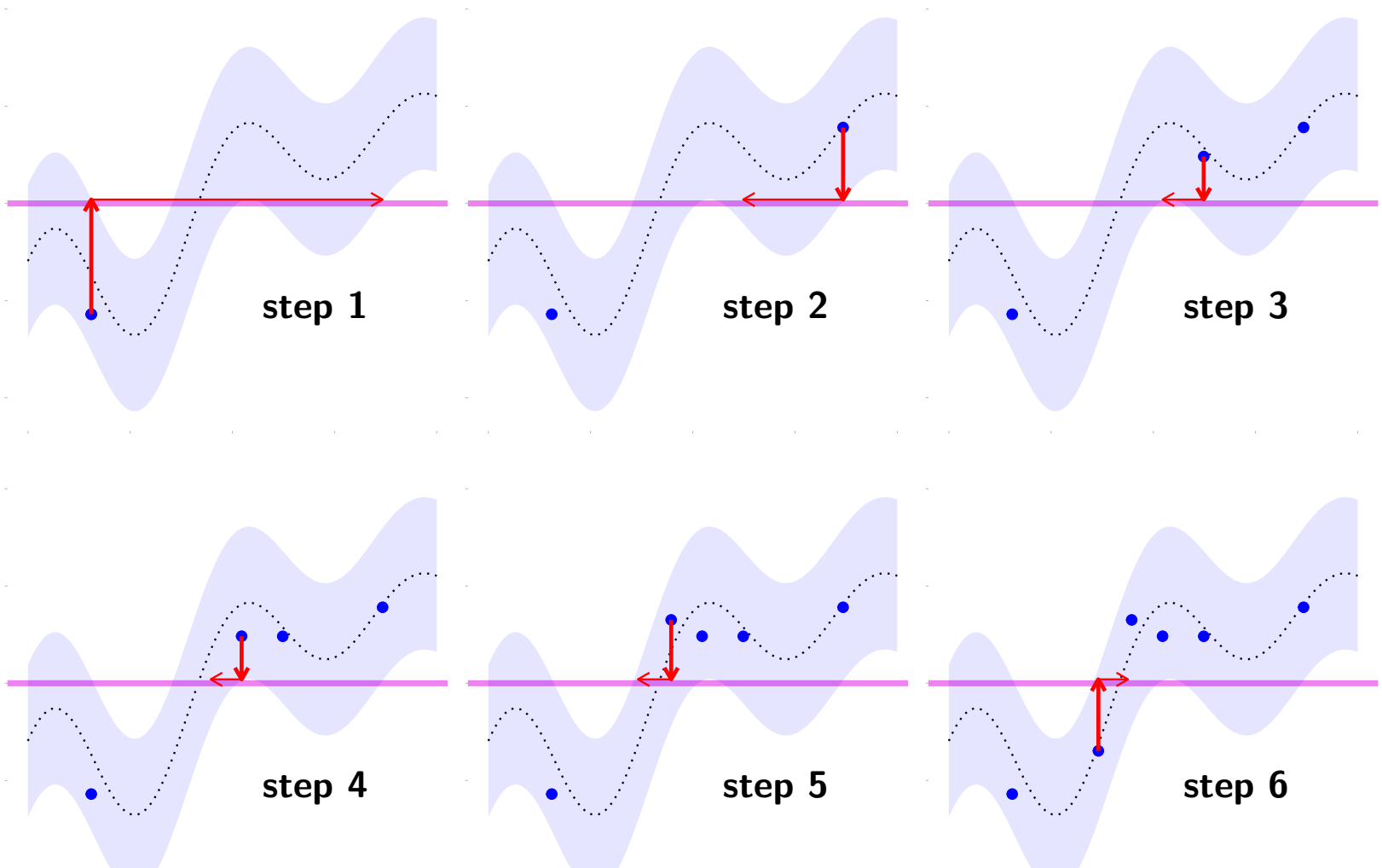
[19] Tieleman, T. and Hinton, G. (2012). Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn*, 17.

[20] Wolfowitz, J. (1952). On the stochastic approximation method of robbins and monro. *The Annals of Mathematical Statistics*, pages 457–461.

6. Appendix

Stochastic approximation - 1D model

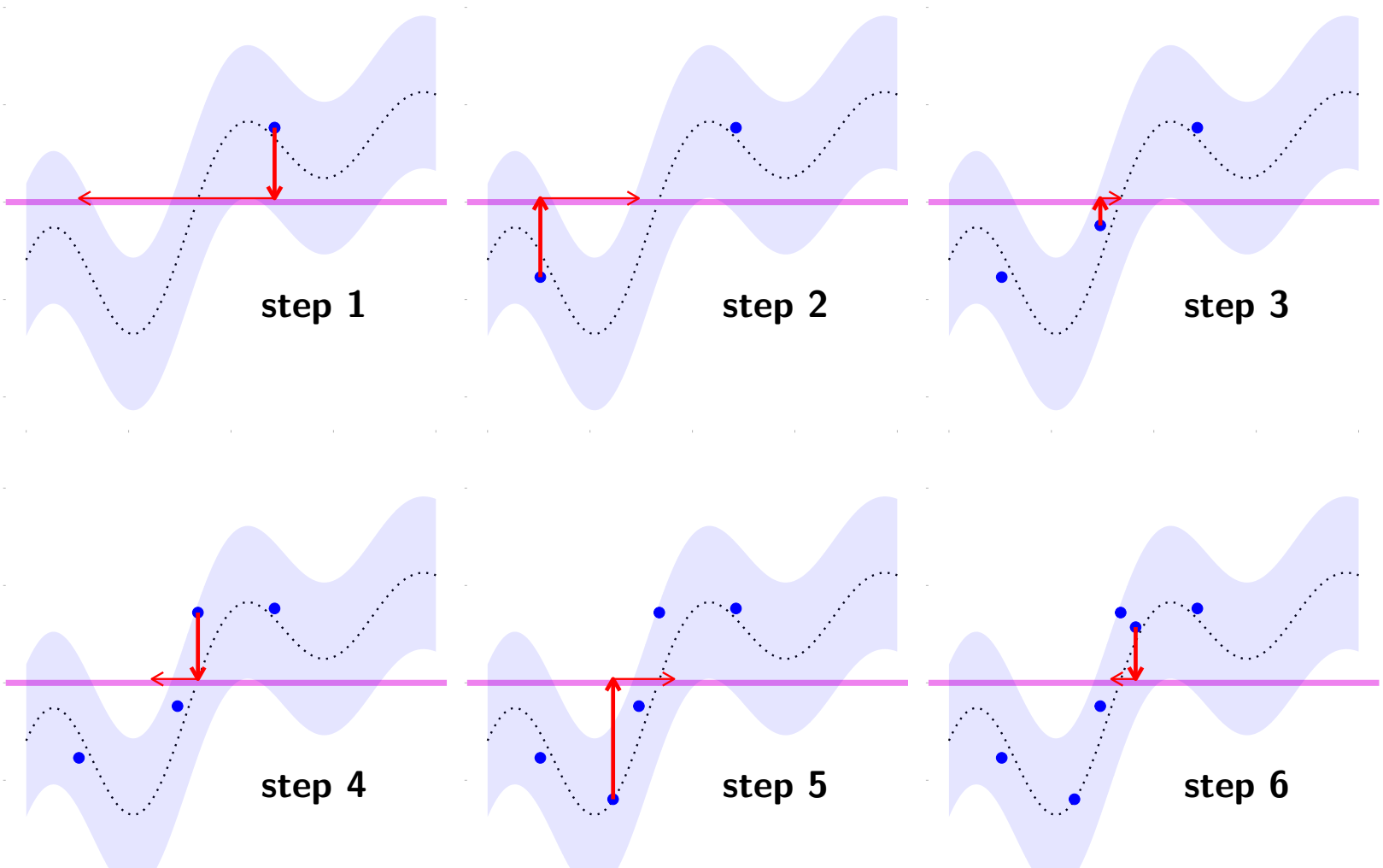
back to main slides illustration of the recursion $\Theta_{n+1} = \Theta_n + a_n (\alpha - F(\Theta_n))$, another seed= 17



Stochastic approximation - 1D model

back to main slides

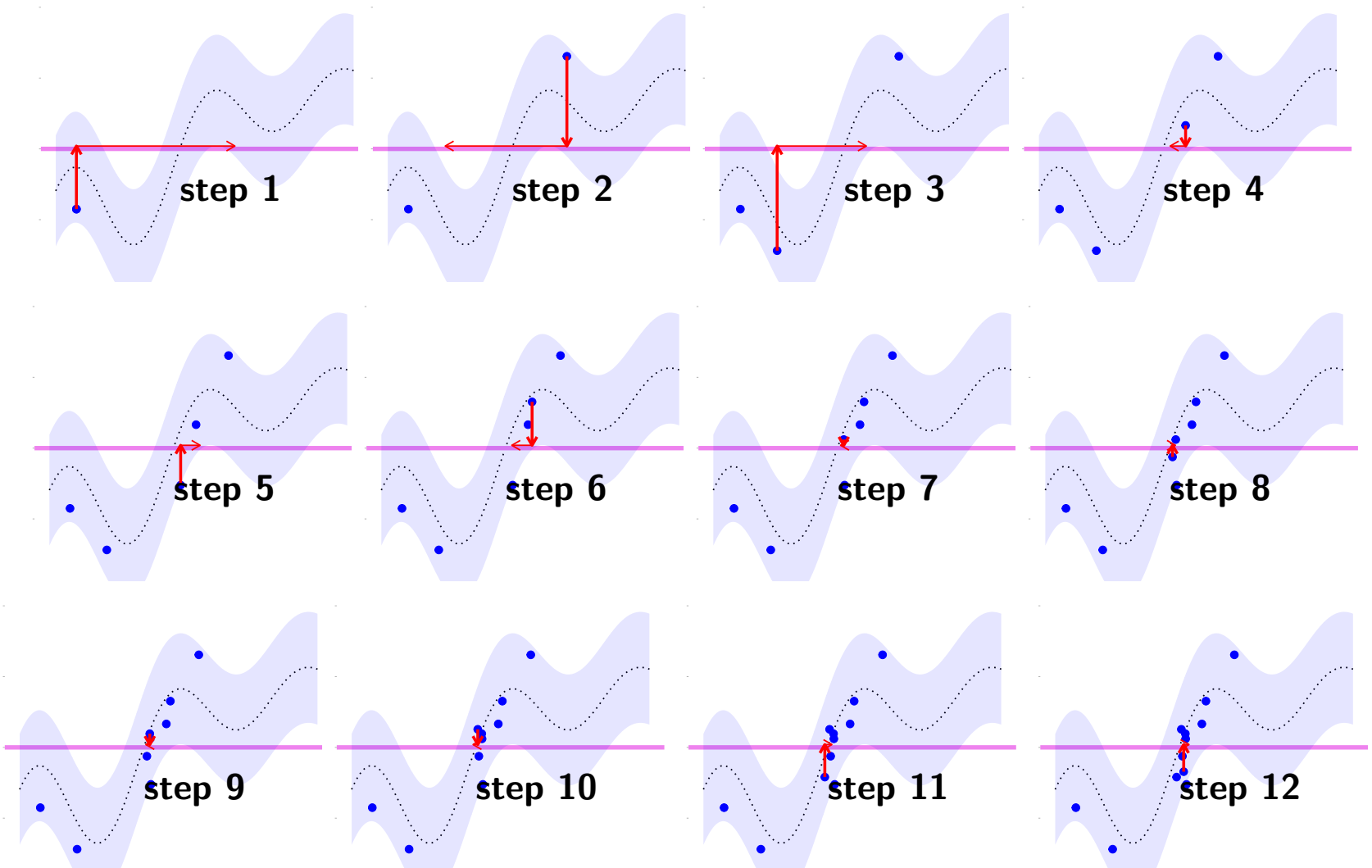
illustration of the recursion $\Theta_{n+1} = \Theta_n + a_n (\alpha - F(\Theta_n))$, another seed$= 6$



Stochastic approximation - 1D model

back to main slides

illustration of the recursion $\Theta_{n+1} = \Theta_n + a_n (\alpha - F(\Theta_n))$, another seed$= 12$



Stochastic Optimization - KW proof (I)

Back to main slides

Step 0: Main quantity to study:

$$\mathrm{E}\left[(\Theta_n - \theta^\star)^2\right] .$$

Applying the recursion (10), $\Theta_{n+1} = \Theta_n - a_n \frac{F(\Theta_n + c_n) - F(\Theta_n - c_n)}{2c_n}$, and denoting

$$\left\{ \begin{array}{rcl} e_n & = & \mathrm{E}\left[\left(\frac{F(\Theta_n + c_n) - F(\Theta_n - c_n)}{2}\right)^2\right] \\ u_n(\theta) & = & (\theta - \theta^\star)\,\mathrm{E}\left[\frac{F(\theta + c_n) - F(\theta - c_n)}{2}\right] \end{array} \right. \tag{28}$$

we get

$$\mathrm{E}\left[(\Theta_{n+1} - \theta^\star)^2\right] = \mathrm{E}\left[(\Theta_n - \theta^\star)^2\right] - 2\frac{a_n}{c_n}\,\mathrm{E}\left[u_n(\Theta_n)\right] + \frac{a_n^2}{c_n^2} e_n \tag{29}$$

hence by summation , with $x_+ = \max(x, 0)$ and $x_- = \min(x, 0)$

$$\mathrm{E}\left[(\Theta_{n+1} - \theta^\star)^2\right] = \mathrm{E}\left[(\Theta_1 - \theta^\star)^2\right] - 2\sum_{j=1}^{n} \frac{a_j}{c_j}\,\mathrm{E}\left[u_j(\Theta_j)_+ + u_j(\Theta_j)_-\right] + \sum_{j=1}^{n} \frac{a_j^2}{c_j^2} e_j$$

... does it converge?

Stochastic gradient descent - KW proof (II)

$$\mathrm{E}\left[\left(\Theta_{n+1}-\theta^\star\right)^2\right] = \mathrm{E}\left[\left(\theta_1-\theta^\star\right)^2\right] - 2\sum_{j=1}^{n}\frac{a_j}{c_j}\,\mathrm{E}\left[u_j(\Theta_j)_+ + u_j(\Theta_j)_-\right] + \sum_{j=1}^{n}\frac{a_j^2}{c_j^2}e_j$$

Back to main slides

Step 1: Show that all sums are finite

- $u_j(\theta) = \frac{1}{2}(\theta-\theta^\star)\{f(\theta+c_j)-f(\theta-c_j)\}$ can only be negative at distance c_j from $\theta^\star$.
- **C1**: near $\theta^\star$ Lipschitz, variation of $f(.)$ not too large $\Longrightarrow \left|u_j(\theta)_-\right| < 2Bc_j^2$

  H2: $\sum a_j c_j < +\infty$:

  $\Longrightarrow \sum_{j=1}^{n}\frac{a_j}{c_j}\,\mathrm{E}\left[\left|u_j(\Theta_j)_-\right|\right] < +\infty$
- **C0**: finite variance, and **C2**: Lipschitz

  $\Longrightarrow e_n = \frac{1}{4}\,\mathrm{E}\left[(F(\Theta_n+c_n)-F(\Theta_n-c_n))^2\right]$ bounded for n large enough

  H3: $\sum a_n^2 c_n^{-2} < +\infty$

  $\Longrightarrow \sum_{j=1}^{n}\frac{a_j^2}{c_j^2}e_j < +\infty$
- $(\theta_1-\theta^\star)^2 - 2\sum_{j=1}^{n}\frac{a_j}{c_j}\,\mathrm{E}\left[u_j(\Theta_j)_+ + u_j(\Theta_j)_-\right] + \sum_{j=1}^{n}\frac{a_j^2}{c_j^2}e_j \geq 0$

  $\Longrightarrow \left|\sum_{i=1}^{n}\frac{a_j}{c_i}\,\mathrm{E}\left[u_j(\Theta_j)_+\right]\right| < +\infty$

Stochastic gradient descent - KW proof (III)

Back to main slides

Step 2: Use finite absolute sums of gradients expectation

- From previous finite sums,
  $\Longrightarrow \sum_{j=1}^{n} \frac{a_j}{c_j} \mathrm{E}\left[|u_j(\Theta_j)|\right] < +\infty$
- **H1**: $\sum a_n = +\infty$,

  $\Longrightarrow \liminf_{n\to+\infty} \mathrm{E}\left[|u_j(\Theta_j)/c_j|\right] = 0$,
  $\Longrightarrow \liminf_{n\to+\infty} \mathrm{E}\left[\left|(\Theta_j - \theta^\star)\frac{f(\Theta_j+c_j)-f(\Theta_j-c_j)}{2c_j}\right|\right] = 0$ (*)
- If $Z_n \underset{n\to+\infty}{\overset{P}{\not\to}} \theta^\star$ then $\exists \delta, \epsilon,\ \mathrm{P}\left[\left|\Theta_{t_j} - \theta^\star\right| > \delta\right] > \epsilon$.

  Then using Markov Inequality $\mathrm{E}[X] \geq a\,\mathrm{P}[X \geq a]$, $\mathrm{E}\left[\left|\Theta_{t_j} - \theta^\star\right|\right] \geq \delta\epsilon$
- **C3**: absolute derivative of $f(.)$ not too small far from $\theta^\star$

  $\Longrightarrow$ contradiction with (*) and thus (skipping details)

$$\Theta_n \underset{n\to+\infty}{\overset{P}{\to}} \theta^\star$$

Back to main slides

Application to Machine Learning - General loss function

back to main slides Where is the random function to optimize?

Consider the following quantities:

- A data, $\mathbf{row}_i$ contains both inputs and (random) outputs at row i.

$$\text{Data} = \{\mathbf{row}_i\}_{i=1,\dots,n}$$

- A loss function indicating the model error for the parameters $\boldsymbol{\theta}$ and a row $\mathbf{row}_i$,

$$\ell(\boldsymbol{\theta}, \mathbf{row}_i)$$

Examples

- loss function $\ell(.)$ in supervised learning: e.g. error between prediction and output
- the model can be a neural network, with $\boldsymbol{\theta}$ containing weights and bias.

For a (random) sample $\mathcal{K} \subseteq \{1, \dots, n_{\text{obs}}\}$ we aim at minimizing the expected loss:

$$f_{\mathcal{K}}(\boldsymbol{\theta}) := \mathrm{E}\left[F_{\mathcal{K}}(\boldsymbol{\theta})\right] \quad \text{with} \quad F_{\mathcal{K}}(\boldsymbol{\theta}) := \frac{1}{\operatorname{card}\mathcal{K}} \sum_{K \in \mathcal{K}} \ell(\boldsymbol{\theta}, \mathbf{row}_K)$$

QUIZZ What makes the observed loss $F_{\mathcal{K}}(\boldsymbol{\theta})$ noisy?

QUIZZ How to adapt this for non-supervised learning, e.g. clustering?

Stochastic gradient descent - A toy example

back to main slides

A simple toy example (I)

The result of a dice is a random variable D.

☺ Your manager wants you to give a value $\theta^\star$ close the next result.

☹ You lose as many euros as you fail on the next guess, i.e. $|D - \theta^\star|$.

1. Model: The loss for a candidate guess θ is a random variable:

$$\ell(\theta, D) = |D - \theta|$$

2. Parameter: Which $\theta^\star$ to propose? e.g. find a $\theta^\star$ that minimizes an expected loss:

$$\theta^\star \in \operatorname*{argmin}_\theta \mathrm{E}\left[\ell(\theta, D)\right] =: \operatorname*{argmin}_\theta f(\theta)$$

3. Data. To help you, the dice-rolling service can throw the dice n times he can give you an iid sequence D_1, D_2,...,$D_n \in \{1, ..., 6\}$ the expected loss $f(\theta) = \mathrm{E}\left[\ell(\theta, D)\right]$ can be estimated by a value,

$$F(\theta) := \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, D_i) = \frac{1}{n}\sum_{i=1}^{n} |D_i - \theta| = f(\theta) + \mathrm{noise}(\theta)$$

but it is still random, can be seen as a noisy evaluation of the true expected loss.

Stochastic gradient descent - Regression Example

back to main slides

A simple linear regression model example (II)

1. Model: Consider a couple of random variables $(\text{Pluie}, \text{Rendement})$.
   One assumes the following link, where $\mathcal{E}$ is a $\mathcal{N}(0, \sigma^2)$ r.v. indep of Pluie,
   $$\text{Rendement} = \beta\,\text{Pluie} + \mathcal{E}$$
   One wants to estimate the value of the unknown parameter β.
2. Parameter A prediction error for a candidate parameter θ, it is a random variable:
   $$\ell(\theta, \text{Pluie}, \text{Rendement}) = (\text{Rendement} - \theta\,\text{Pluie})^2$$
   Let β^* minimizes the expected loss between predictions and observations.
   $$\beta^* = \operatorname*{argmin}_\theta \mathrm{E}\left[\ell(\theta, \text{Pluie}, \text{Rendement})\right] =: \operatorname*{argmin}_\theta f(\theta) \quad \text{...show } \beta^* = \beta.$$
3. Data Given an iid sample of $(\text{Pluie}, \text{Rendement})$,

pluie	rendement	
0.0556	0.1316	$\leftarrow \mathbf{row}_1 = (\text{Pluie}_1, \text{Rendement}_1)$
0.0623	0.0753	$\leftarrow \mathbf{row}_2 = (\text{Pluie}_2, \text{Rendement}_2)$
...	...	

one only observes noisy estimations of $f(x)$ (even with *full batch* here)

$$F(\theta) := \frac{1}{n}\sum^{n} \ell(\theta, \mathbf{row}_i) = f(\theta) + \text{noise}(\theta)$$

Stochastic Gradient Descent - AI example

back to main slides

1. Model. An Artificial Intelligence tries to predict prices from images of objects.

$$\text{price} = h(\textbf{image}, \theta^\star) + \mathcal{E}$$

where h is a given function (neural network) and $\mathcal{E}$ is a centered random variable. One wants to estimate the value of the unknown parameter vector $\theta^\star$.

2. Loss function. Treatment of images is long, one pick one data row K at random: $\textbf{row}_K = (\textbf{image}_K, \text{price}_K)$. A prediction error for a candidate parameter θ:

$$\ell(\theta, \textbf{row}_K) = (\text{price}_K - h(\textbf{image}_K, \theta))^2$$

This is a random variable: one pick data at random, and model is not perfect ($\mathcal{E}$).

3. Parameter. Let $\theta^\star$ minimizes the expected loss between prediction and observation.

$$\theta^\star = \operatorname*{argmin}_\theta \mathrm{E}\left[\ell(\theta, \textbf{row}_K)\right] =: \operatorname*{argmin}_\theta f(\theta)$$

for a parameter θ one only observes noisy estimations of $f(\theta)$:

$$F(\theta) := \ell(\theta, \textbf{row}_K) \quad \text{with} \quad f(\theta) := \mathrm{E}\left[F(\theta)\right]$$

QUIZZ Show $f(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(\theta, \textbf{row}_i)$ if K is uniform and $\mathcal{E} = 0$

Application to Machine Learning - Neural Net example

back to main slides

One practical example of a loss function on Neural Network

The neural network model relies on:

- a data $\mathrm{Data} = \{\mathbf{row}_i\}_{i=1,\dots,n_{\mathrm{obs}}} = \{(\mathbf{x}_i, Y_i)\}_{i=1,\dots,n_{\mathrm{obs}}}$
- a function $h = \mathbf{h}_1 \circ \dots \circ \mathbf{h}_k$, depending on weights and bias θ, here for k-layers
- outputs Y_i are close to the model value $h(\mathbf{x}_i; \theta^\star)$, with random noises $\mathcal{E}(\mathbf{x}_i)$,

$$Y_i = h(\mathbf{x}_i; \theta^\star) + \mathcal{E}(\mathbf{x}_i)\,.$$

An example of possible resulting noisy loss for a random sample $\mathcal{K} = \{K\}$:

$$\ell(\theta, \mathbf{row}_K) = (h(\mathbf{x}_K; \theta) - Y_K)^2$$

.. and with respect to parameters θ, the gradient of the expected loss is computable

$$\nabla f(\theta) := \nabla\, \mathrm{E}\left[\ell(\theta, \mathbf{row}_K)\right] = \mathrm{E}\left[\nabla\left(h(\mathbf{x}_K; \theta) - Y_K\right)^2\right] =: \mathrm{E}\left[\widehat{\mathbf{G}}_K(\theta)\right]$$

Details on gradient backpropagation

Gradient backpropagation (i)

Back to main slides

Univariate case: understanding the scheme

- Consider real-valued functions f and g, parameters $\lambda, \mu \in \mathbb{R}$, and $x, y, z \in \mathbb{R}$

$$\begin{array}{ccccc} & & \overset{f(x,\lambda)}{\overset{\shortparallel}{y}} & & \overset{g(y,\mu)=g(f(x,\lambda),\mu)}{\overset{\shortparallel}{z}} \\ x & \underset{\leftarrow}{\rightarrow} & & \underset{\leftarrow}{\rightarrow} & \end{array}$$

To get the variation of z with respect to parameters (λ, μ) we need:

$$\begin{cases} \frac{\partial z}{\partial \mu} &= \text{direct calculation} \\ \frac{\partial z}{\partial \lambda} &= \frac{\partial z}{\partial y} \times \frac{\partial y}{\partial \lambda} \quad = \frac{\partial g(y,\mu)}{\partial y} \frac{\partial f(x,\lambda)}{\partial \lambda} = (g' \circ f) \cdot f' \end{cases} \tag{30}$$

key point: need to *back propagate* the sensitivity to inputs $\frac{\partial z}{\partial y}$ by chain rule.

- Iterative scheme for a loss function ℓ, for parameter θ_k, at layer k:

$$\begin{array}{ccccccc} & & \overset{f(x_{k-1},\theta_k)}{\overset{\shortparallel}{x_k}} & & & & \\ x_{k-1} & \underset{\leftarrow}{\rightarrow} & & \underset{\leftarrow}{\rightarrow} & \dots & \underset{\leftarrow}{\rightarrow} & \ell \end{array}$$

$$\begin{cases} \frac{\partial \ell}{\partial \theta_k} &= \frac{\partial \ell}{\partial x_k} \times \frac{\partial x_k}{\partial \theta_k} \\ \frac{\partial \ell}{\partial x_{k-1}} &= \frac{\partial \ell}{\partial x_k} \times \frac{\partial x_k}{\partial x_{k-1}} \end{cases} \tag{31}$$

Gradient backpropagation (ii)

Back to main slides

Multivariate case: understanding the calculations

- Recall previous slide, iterative scheme for a loss function ℓ, at layer k:

$$\begin{array}{ccccccccc} & & & & f(x_{k-1},\theta_k) & & & & \\ & & & & \| & & & & \\ x_{k-1} & \underset{\leftarrow}{\rightarrow} & & & x_k & \underset{\leftarrow}{\rightarrow} & \dots & \underset{\leftarrow}{\rightarrow} & \ell \end{array}$$

$$\begin{cases} \dfrac{\partial \ell}{\partial \theta_k} &= \dfrac{\partial \ell}{\partial x_k} \times \dfrac{\partial x_k}{\partial \theta_k} \\ \dfrac{\partial \ell}{\partial x_{k-1}} &= \dfrac{\partial \ell}{\partial x_k} \times \dfrac{\partial x_k}{\partial x_{k-1}} \end{cases} \tag{31}$$

- Multivariate adaptation: (horizontal gradients here)

$$\begin{cases} \dfrac{\partial \ell}{\partial \boldsymbol{\theta}_k} &= \dfrac{\partial \ell}{\partial \mathbf{x}_k} \times \left(\dfrac{\partial \mathbf{x}_k}{\partial \boldsymbol{\theta}_k}\right) \\ \dfrac{\partial \ell}{\partial \mathbf{x}_{k-1}} &= \dfrac{\partial \ell}{\partial \mathbf{x}_k} \times \left(\dfrac{\partial \mathbf{x}_k}{\partial \mathbf{x}_{k-1}}\right) \end{cases} \tag{32}$$

where $\left(\dfrac{\partial \mathbf{x}_k}{\partial \boldsymbol{\theta}_k}\right)$ and $\left(\dfrac{\partial \mathbf{x}_k}{\partial \mathbf{x}_{k-1}}\right)$ are Jacobian matrices, other quantities are gradients.

more details

https://www.college-de-france.fr/site/stephane-mallat/course-2019-03-20-09h30.htm

Gradient backpropagation (iii)

Back to main slides

Multivariate case: calculations in a specific case

For a weights matrix W_k and an element-wise function $\sigma_k(.)$, assume at layer k

$$(\mathrm{H}_k): \quad \mathbf{x}_k = \sigma_k(W_k \mathbf{x}_{k-1} + \mathbf{b}_k)$$

Then under (H_k), defining the horizontal vector $\mathbf{v}_k := \frac{\partial \ell}{\partial \mathbf{x}_k} \mathrm{diag}\left[\sigma_k'\left(W_k \mathbf{x}_{k-1} + \mathbf{b}_k\right)\right]$,

$$\text{(please check!)} \quad \begin{cases} \left(\frac{\partial \ell}{\partial W_k}\right) &= \mathbf{v}_k{}^\top \mathbf{x}_{k-1}{}^\top \\ \left(\frac{\partial \ell}{\partial \mathbf{b}_k}\right) &= \mathbf{v}_k \\ \frac{\partial \ell}{\partial \mathbf{x}_{k-1}} &= \mathbf{v}_k W_k \end{cases} \tag{33}$$

Gradient backpropagation (iv)

Back to main slides

Multivariate case: calculations in a specific case, proof

Proof: let $\mathbf{y} := \sigma(W\mathbf{x} + \mathbf{b})$. Denoting $\mathbf{x}^+ := W\mathbf{x} + \mathbf{b}$, $(n \times 1) = (n \times m)(m \times 1) + (n \times 1)$

- $\frac{\partial y_i}{\partial x_j} = \frac{\partial \sigma(W_{i.}\mathbf{x} + b_i)}{\partial \mathbf{x}_j} = \sigma'(\mathbf{x}_i^+) W_{ij}$ hence $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right) = \operatorname{diag}\left[\sigma'(\mathbf{x}^+)\right] W = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}^+}\right)\left(\frac{\partial \mathbf{x}^+}{\partial \mathbf{x}}\right)$

$$\frac{\partial \ell}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \mathbf{y}} \times \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right) = \frac{\partial \ell}{\partial \mathbf{y}} \operatorname{diag}\left[\sigma'(\mathbf{x}^+)\right] W \quad (1 \times m) = (1 \times n)(n \times n)(n \times m)$$

- $\frac{\partial y_i}{\partial b_j} = \frac{\partial \sigma(W_{i.}\mathbf{x} + b_i)}{\partial b_j} = \mathbb{1}_{\{i=j\}} \sigma'(\mathbf{x}_i^+)$, and

$$\frac{\partial \ell}{\partial \mathbf{b}} = \frac{\partial \ell}{\partial \mathbf{y}} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{b}}\right) = \frac{\partial \ell}{\partial \mathbf{y}} \operatorname{diag}\left[\sigma'(\mathbf{x}^+)\right] \quad (1 \times n) = (1 \times n)(n \times n)$$

- $\frac{\partial y_{i'}}{\partial W_{ij}} = \frac{\partial \sigma(W_{i'.}\mathbf{x} + b_{i'})}{\partial W_{ij}} = \mathbb{1}_{\{i=i'\}} \sigma'(\mathbf{x}_i^+) x_j$. Hence for a vector $\boldsymbol{\theta} = (\theta_{[ij]}) = (W_{ij})$,

$$\left(\frac{\partial \ell}{\partial \boldsymbol{\theta}}\right)_{[ij]} = \sum_{i'=1}^{n} \left(\frac{\partial \ell}{\partial \mathbf{y}}\right)_{i'} \frac{\partial y_{i'}}{\partial \theta_{[ij]}} = \left(\frac{\partial \ell}{\partial \mathbf{y}}\right)_i \sigma'(\mathbf{x}_i^+) x_j$$

$$\text{and} \quad \left(\frac{\partial \ell}{\partial W}\right) = \operatorname{diag}\left[\sigma'(\mathbf{x}^+)\right] \left(\frac{\partial \ell}{\partial \mathbf{y}}\right)^\top \mathbf{x}^\top \quad (n \times m) = (n \times n)(n \times 1)(1 \times m)$$

other sol. $\frac{\partial \ell}{\partial W_{ij}} = \left(\frac{\partial \ell}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}^+}\right) \frac{\partial \mathbf{x}^+}{\partial W_{ij}} = v_i x_j$ with $\mathbf{v} := \frac{\partial \ell}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}^+} = \frac{\partial \ell}{\partial \mathbf{y}} \operatorname{diag}\left[\sigma'(\mathbf{x}^+)\right]$.

Hint on vector derivatives

Back to main slides Vector derivatives

How are arranged elements of a Jacobian matrice, is it transposed or not?
For clearer notation conventions, conventions must hold for scalar 1×1 vectors.

Consider two column vectors $\mathbf{u}$ and $\mathbf{v}$ of sizes p and q, and a scalar α.

- The dimension of $\frac{\partial \mathbf{u}}{\partial \alpha}$ is clearly $p \times 1$
- Hence the dimension of the Jacobian $\left(\frac{\partial \mathbf{u}}{\partial \mathbf{v}}\right) = \left(\frac{\partial u_i}{\partial v_j}\right)_{i,j}$ is logically $p \times q$
- and the dimension of $\frac{\partial \alpha}{\partial \mathbf{v}}$ is $1 \times q$ (e.g. horizontal gradients)

This is the same for row vectors $u^\top$ and $v^\top$:

- The dimension of $\frac{\partial \mathbf{u}^\top}{\partial \alpha}$ is clearly $1 \times p$
- Hence the dimension of $\frac{\partial \mathbf{u}^\top}{\partial \mathbf{v}^\top}$ is logically $q \times p$

more details on vector and matrices derivatives: `https://cs231n.stanford.edu/vecDerivs.pdf`

On this document, 2024-2025

Version of this document: December 1, 2025.

This document has been slightly modified compared to the one presented previous year. Some notations have changed, we are working at unifying them among different lectures.

- a summary of all notations has been added, see last slide ☺
- we have opted here for usual neural notations, optimizing f as a function of θ
- more details on acceleration techniques
- the organization has changed a little bit

Typos may remain, if you find one, please mention it here: drulliere@emse.fr

Notations

Optimisation (vectors in bold font)

$\theta, \boldsymbol{\theta}$	abscissa of studied function
$\Theta, \boldsymbol{\Theta}$	random abscissa
$\theta^\star, \boldsymbol{\theta}^\star$	minimiser or root
$\mathcal{E}_\theta$, $\mathcal{E}_{\boldsymbol{\theta}}$	a random noise, centered real r.v.
$g(.)$	a function when we search a root
$G(.)$	noisy observations of $g(.)$
α	target threshold when searching root
$f(.)$	a function to minimize, deterministic
$F(.)$	noisy observations of $f(.)$
μ, L, B	strong convexity/smoothness coeffs.

Gradient Descent

$\widehat{G}, \widehat{\mathbf{G}}$	a noisy estimator of gradient
a, a_n	step of gradient descent
c, c_n	step of finite difference
α, γ	decreas. rates $a_n = a/n^\alpha$, $c_n = c/n^\gamma$
β, β_n	momentum coefficient
$\mathbf{M}$	momentum vector
$\mathbf{V}$	second order momentum vector
$\boldsymbol{\Delta}$	random direction vector
v_{ball}	noise ball asymptotic variance
$u(.)$	an intermediary function in KW
$\delta, \pi(\delta)$	some real constants in proofs
η, ϵ, C	other real constants in proofs

Data

$\mathbf{row}$	a row of data (input, output)
Data	whole dataset $= \{\mathbf{row}_i\}_{i=1,\dots,n_{\mathrm{obs}}}$
n_{obs}	number of observations in Data
ℓ	a loss function
$\mathcal{K}$	random subset of indexes in data
K	an element of $\mathcal{K}$, a random row index
D	the result of a dice

Neural network (NN)

$\sigma(.)$	an activation function
$h(.)$	NN function, NN output $= h(input)$
$\mathbf{h}(.)$	vector intermediates functions of NN
$\mathbf{v}, \mathbf{v}_i$	intermediate vector, component
$\mathbf{b}, b_i$	bias vector and component
W, W_{ij}	weights matrix and component
$\boldsymbol{\theta}, \theta_i$	parameter vector, component
$[ij]$	vector index in bijec. with couple (i, j)
k	index of a layer
$\mathbf{x}, x_i$	entry of NN
$\mathbf{x}_k$	signal vector for layer k
$\mathbf{x}_k^+$	vector after transformation $W\mathbf{x}_k + \mathbf{b}$
$\mathbf{y}, y_i$	output of NN
Y	a component of NN output, if random
x, y, z	three successive layers