Chapter 3: Monte Carlo Optimization

UE Computational statistics

Pierre Pudlo

Aix-Marseille Université / Faculté des Sciences

About this chapter

Objectives:

- Understand the principles of Monte Carlo optimization

Content:

- Simulated annealing as an exploration algorithm
- Stochastic gradient descent as an approximation algorithm

> ⚠ **Warning**
>
> This chapter is **much more technical and difficult** than the previous ones.

1. The Problem

Solve the following optimization problem:

$$\underset{\theta \in \Theta}{\operatorname{argmin}}\, h(\theta)$$

Deterministic algorithms:

- Gradient-based methods: Gradient descent, etc.
- Hessian-based methods: Newton, BFGS, etc.

Why Monte Carlo?

1. Exploratory

- The function h may be not differentiable
- Avoid being trapped in local optima

$\Longrightarrow$ simulated annealing, genetic algorithms, etc.

2. Approximation

- The function h may be too costly to evaluate
- In this case, do not even think of computing its gradient

$\Longrightarrow$ Expectation-Maximization (EM) algorithm, stochastic gradient descent, etc.

2. Simulated annealing

- Intend to find the global minimum of h
- Do not resort on the gradient of h
- Its name comes from metallurgy: slowly cooling a metal to get specific physical properties
- Dates back to Metropolis et al. (1953)
- Can solve combinatorial problems such the salesman problem: find the shortest path visiting n cities

2.1 Gibbs measures

Gibbs measures

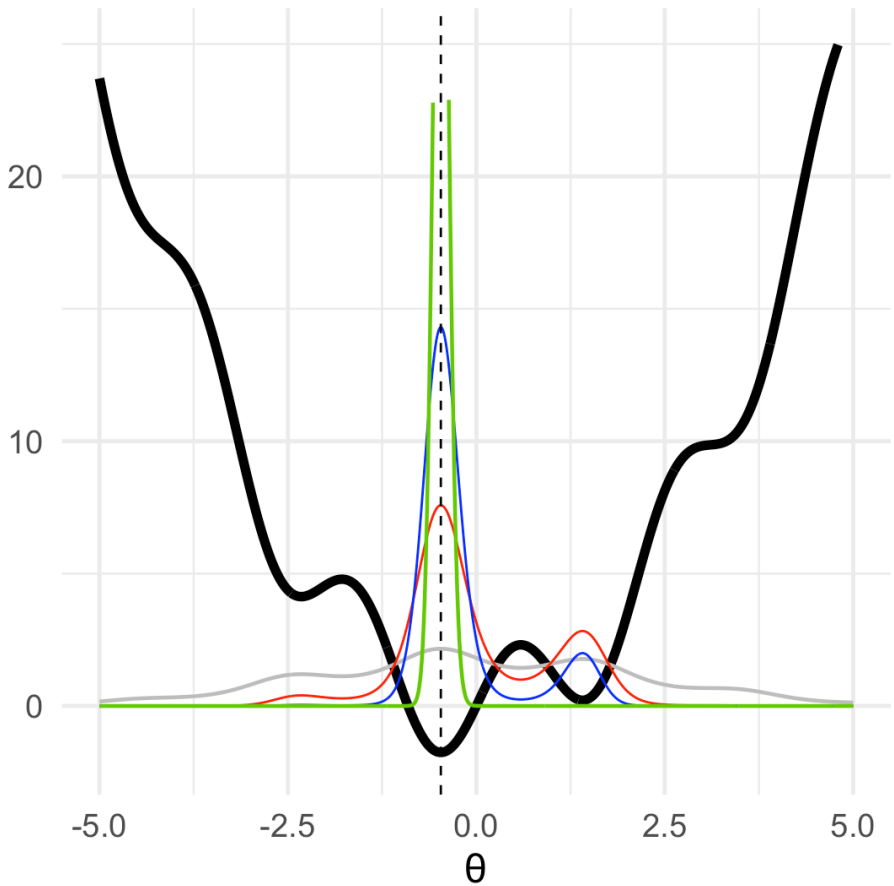
The Gibbs measure of inverse temperature β is a probability measure on Θ defined by

$$\pi_\beta(\theta) \propto \exp\{-\beta h(\theta)\}.$$

Under mild conditions on h, when $\beta \to \infty$, $\pi_\beta(\theta)$ concentrates on the set $\operatorname{argmin}_{\theta \in \Theta} h(\theta)$.

We will not prove this result in this course.

Graphical illustration



Gray : β=1/10, Red : β=1/2, Blue : β=1, Green : β=5

- The black, thick curve: $h(\theta) = \theta^2 + 2\sin(3\theta)$
- The colored curves represent the Gibbs densities: $\pi_\beta(\theta) = \frac{1}{Z(\beta)}\exp\{-\beta h(\theta)\}$
- Local optima disappear as β increases
- The Gibbs measure concentrates on the global minimum

2.2 How to simulate according to a Gibbs measure?

Markov chain Monte Carlo (MCMC) Algorithm

1. Initialize θ_0
2. For $t = 1, 2, \ldots, T$:
   - Generate $\zeta \sim g(\zeta|\theta_t)$, a proposal kernel given θ_t
   - Compute the acceptance ratio $\rho = \dfrac{\exp\{-\beta h(\zeta)\} g(\theta_t|\zeta)}{\exp\{-\beta h(\theta_t)\} g(\zeta|\theta_t)}$
   - Draw $U \sim \mathrm{Unif}(0,1)$ and set $\theta_{t+1} = \zeta \mathbf{1}\{U < \rho\} + \theta_t \mathbf{1}\{U \geq \rho\}$

Explanations

- The **initial value** θ_0 is arbitrary
  - Ideally, it should be chosen in the zone where the Gibbs measure concentrates
- The event $\{U < \rho\}$ is of probability $\rho \wedge 1$ (i.e., the minimum of ρ and 1)
- Given the **current value** θ_t, the algorithm
  - **proposes a new value** ζ according to the proposal kernel g
  - **accepts** ζ with probability $\rho \wedge 1$, **or stays** at θ_t
- The **proposal kernel** g is a conditional distribution:
  - for all $\theta \in \Theta$, $g(\cdot|\theta)$ is a PDF/probability measure
  - the proposal kernel g as to be **chosen carefully** by the user
  - if $\theta \in \mathbb{R}^d$, g is often a **Gaussian kernel**: $[\zeta|\theta_t] \sim \mathcal{N}(\theta_t, \Sigma)$, with a tuned Σ

- The simulated $\theta_1, \ldots, \theta_T$ are **correlated**, and form a **Markov chain**
- The Gibbs measure π_β is **approximated by the empirical measure**:

$$\pi_\beta \approx \frac{1}{T} \sum_{t=1}^{T} \delta_{\theta_t}$$

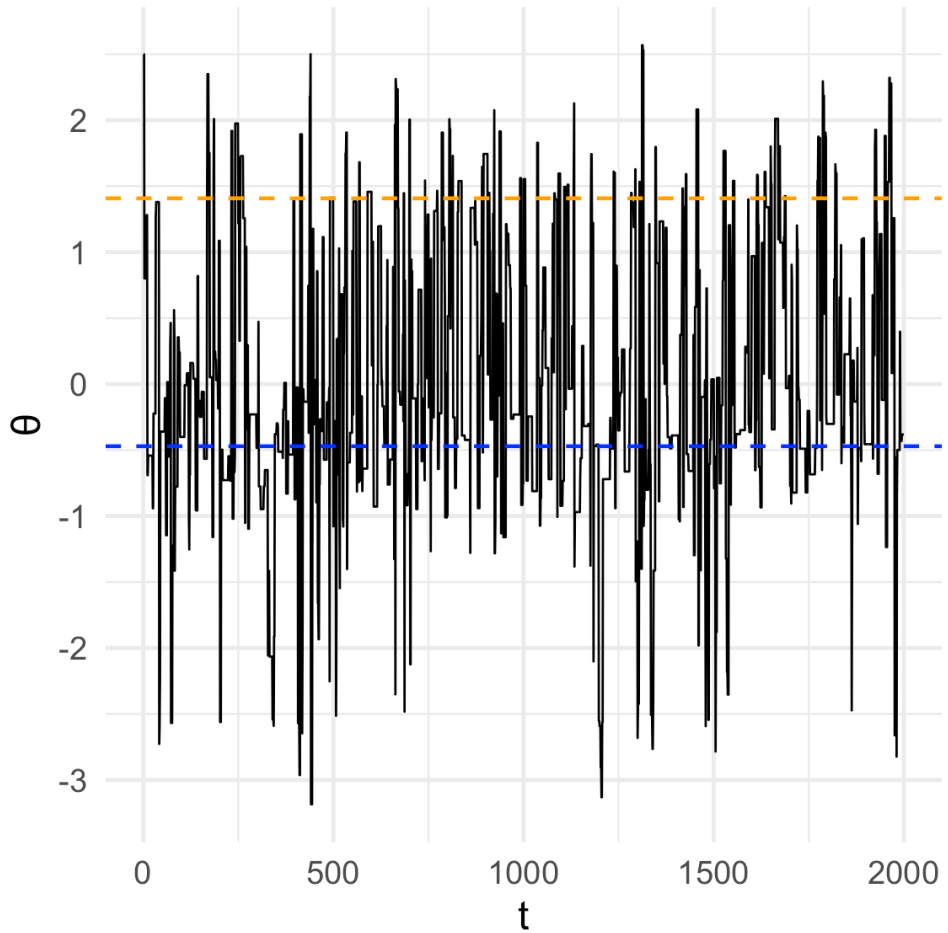
  - i.e, for a large class of functions h,

$$\int h(\theta) \pi_\beta(\theta) \approx \frac{1}{T} \sum_{t=1}^{T} h(\theta_t)$$

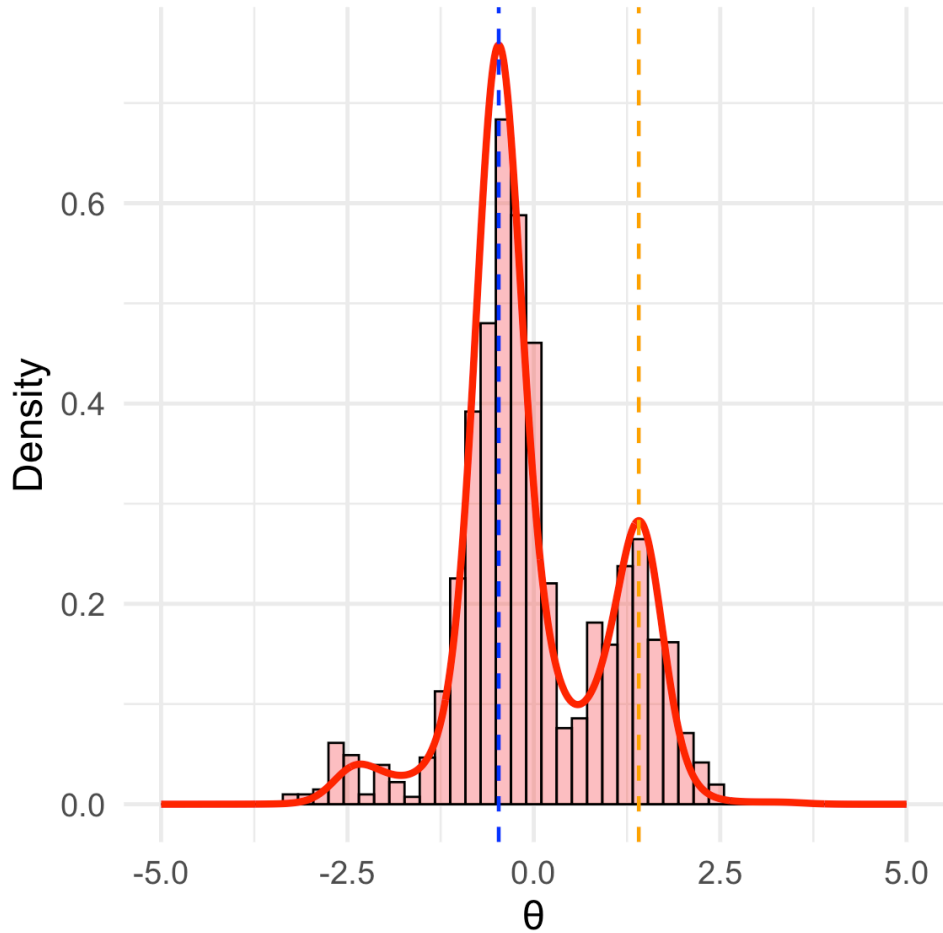
- The **acceptance ratio** ρ is a ratio of Importance Sampling (IS) weights:
  - the proposed ζ is drawn from $g(\cdot|\theta_t)$ and the target is π_β
  - its IS weight would be $w_1 = \exp\{-\beta h(\zeta)\} \Big/ Z(\beta) g(\zeta|\theta_t)$
  - if we switch the role of ζ and θ_t (i.e., $\approx$ backward in time), the IS weight of θ_t would be $w_2 = \exp\{-\beta h(\theta_t)\} \Big/ Z(\beta) g(\theta_t|\zeta)$
  - $\rho = w_1/w_2$; the values of $Z(\beta)$ cancel out in the ratio
- If g is a **symmetric kernel**, i.e. if $g(\zeta|\theta_t) = g(\theta_t|\zeta)$,
  - the acceptance ratio ρ is simpler: $\rho = \exp\{-\beta(h(\zeta) - h(\theta_t))\}$
  - If the proposed ζ do not increase h, i.e., if $h(\zeta) \leq h(\theta_t)$, then $\rho \geq 1$ and $\theta_{t+1} = \zeta$ with probability 1
  - If the ζ increases much h, i.e., if $h(\zeta) \gg h(\theta_t)$, then $\rho \ll 1$ and $\theta_{t+1} = \theta_t$ with high probab.

Numerical examples: Gibbs with $\beta = 1/2$

Trajectory:



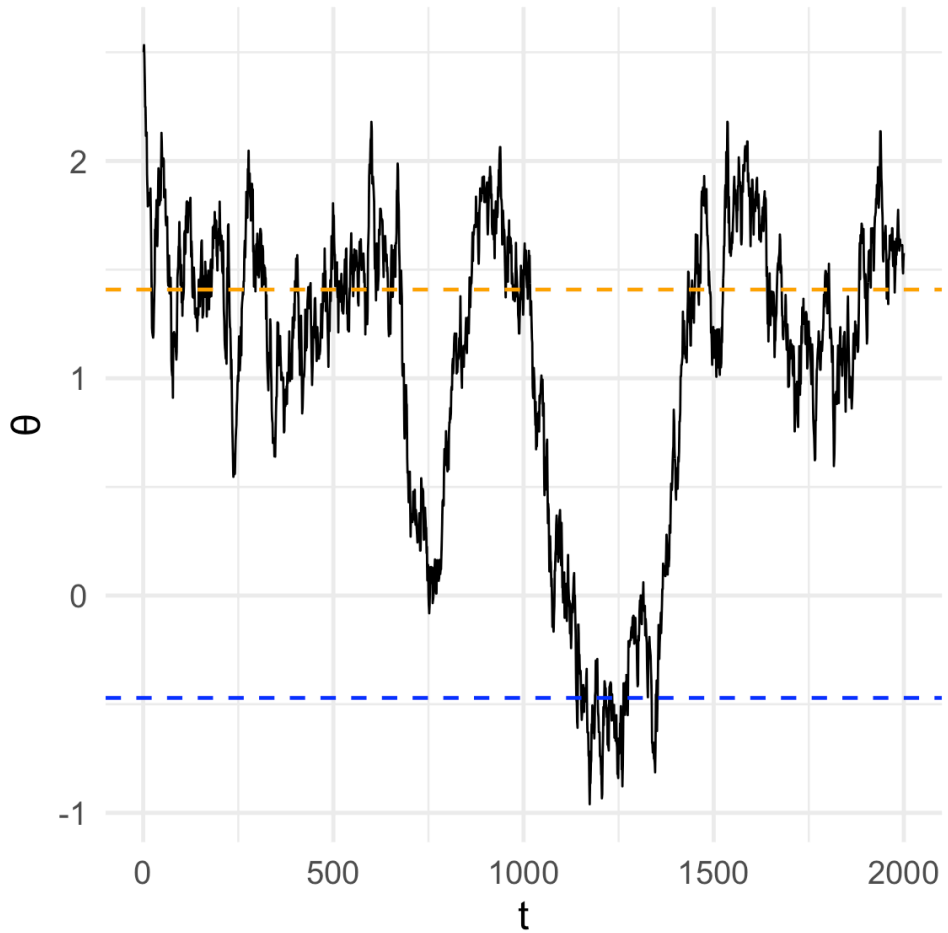
Distrib. of the path & Gibbs density:



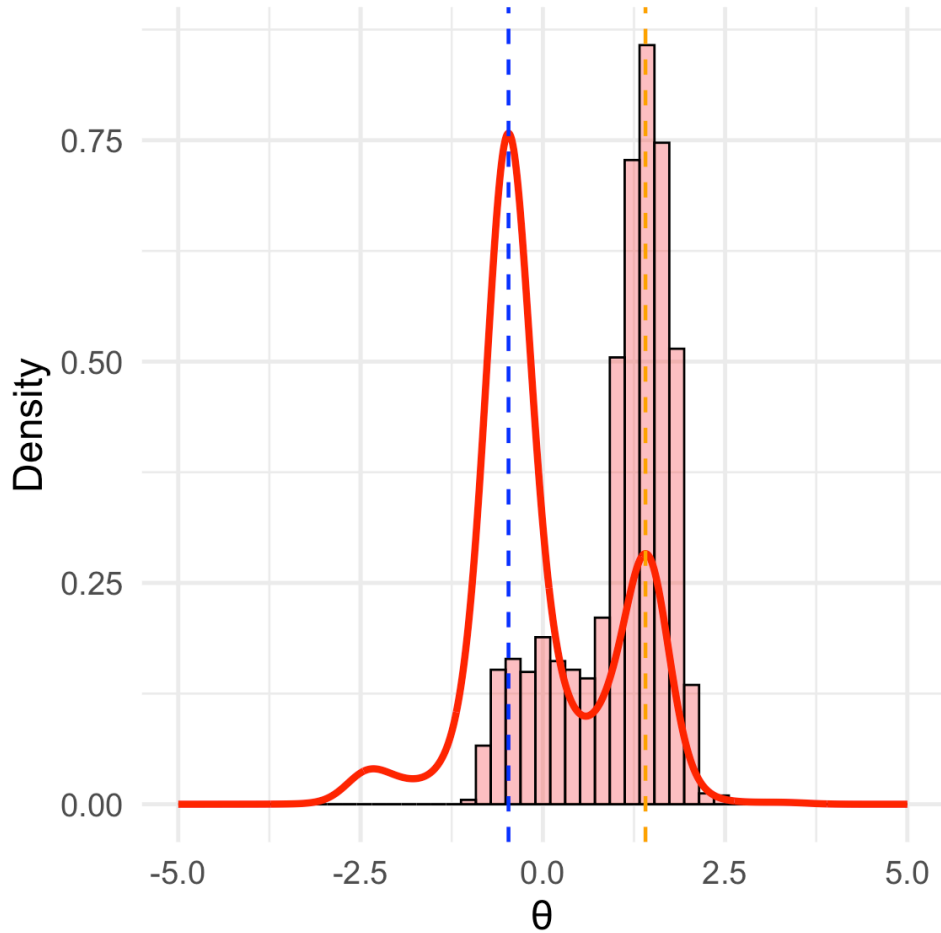
Proposal: $\zeta|\theta_t \sim \mathcal{N}(\theta_t, 3^2)$

Bad tuning of the proposal kernel

Trajectory:



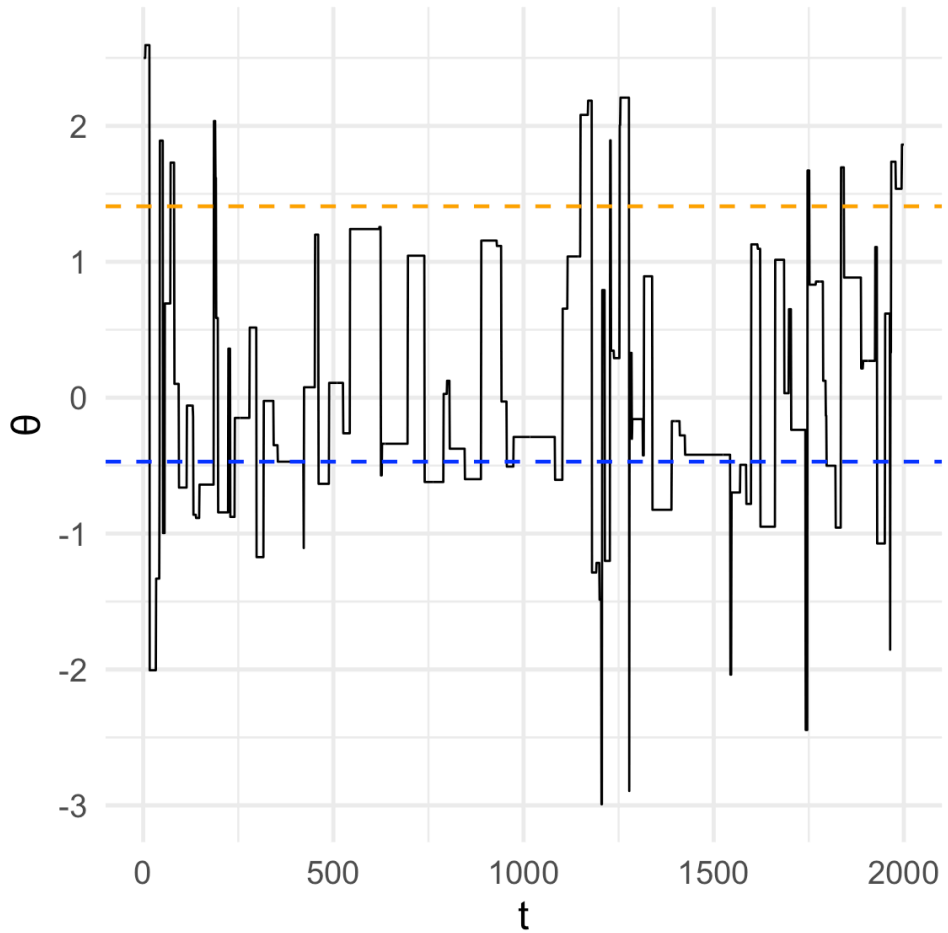
Distrib. of the path & Gibbs density:



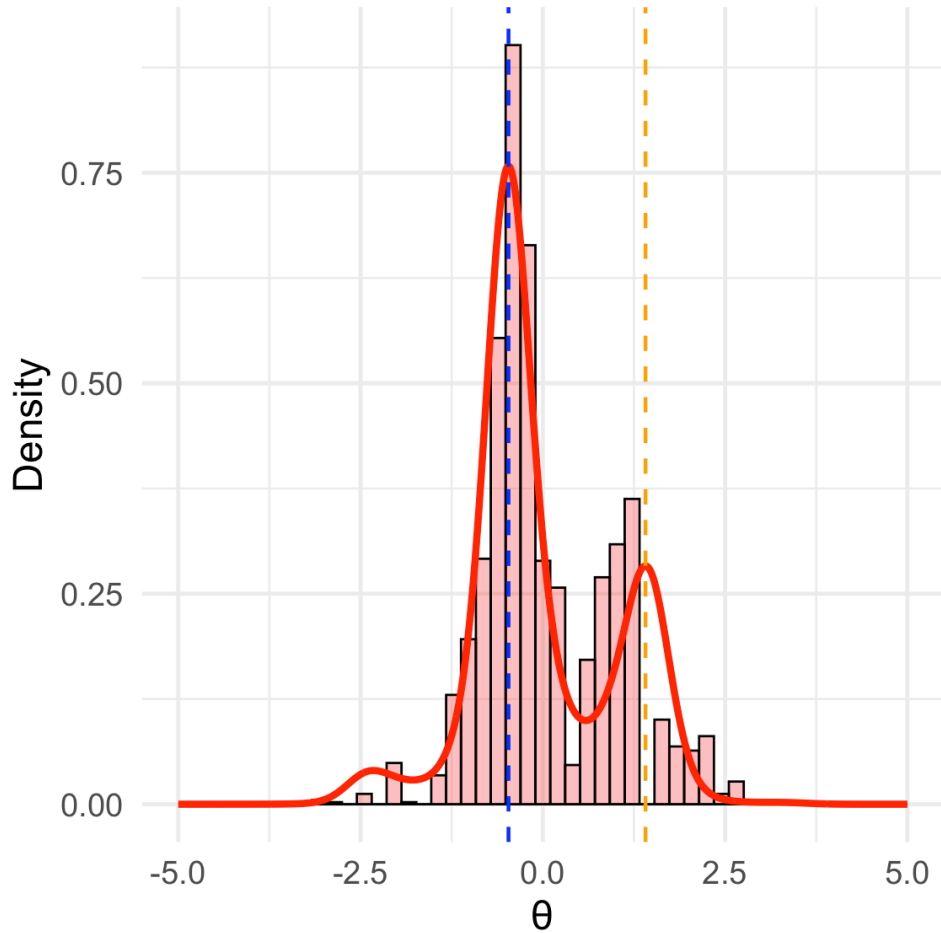
Proposal: $\zeta|\theta_t \sim \mathcal{N}(\theta_t, 0.1^2)$ $\Longrightarrow$ variance too small

Another bad tuning of the proposal kernel

Trajectory:



Distrib. of the path & Gibbs density:



Proposal: $\zeta|\theta_t \sim \mathcal{N}(\theta_t, 20^2)$ $\Longrightarrow$ variance too large

How to tune Σ?

- When the variance is too small,
  - the chain is slow to explore the space
  - $\zeta \approx \theta_t$ and thus $\rho \wedge 1 \approx 1$
- When the variance is too large,
  - the proposed ζ is too far away from the zone of interest: $h(\zeta) \gg h(\theta_t)$
  - the chain moves rarely, $\rho \wedge 1 \approx 0$

Rule

When Σ is tuned correctly, the average value of acceptance probability $\rho \wedge 1$ along the path should be around 0.234.

Numerical examples: (cont'd)

- When $\Sigma = 3^2$, the average value of $\rho \wedge 1$ is

```
0.290467
```

- When $\Sigma = 0.1^2$, the average value of $\rho \wedge 1$ is

```
0.9292316
```

- When $\Sigma = 20^2$, the average value of $\rho \wedge 1$ is

```
0.05161779
```

2.3 Cooling the temperature

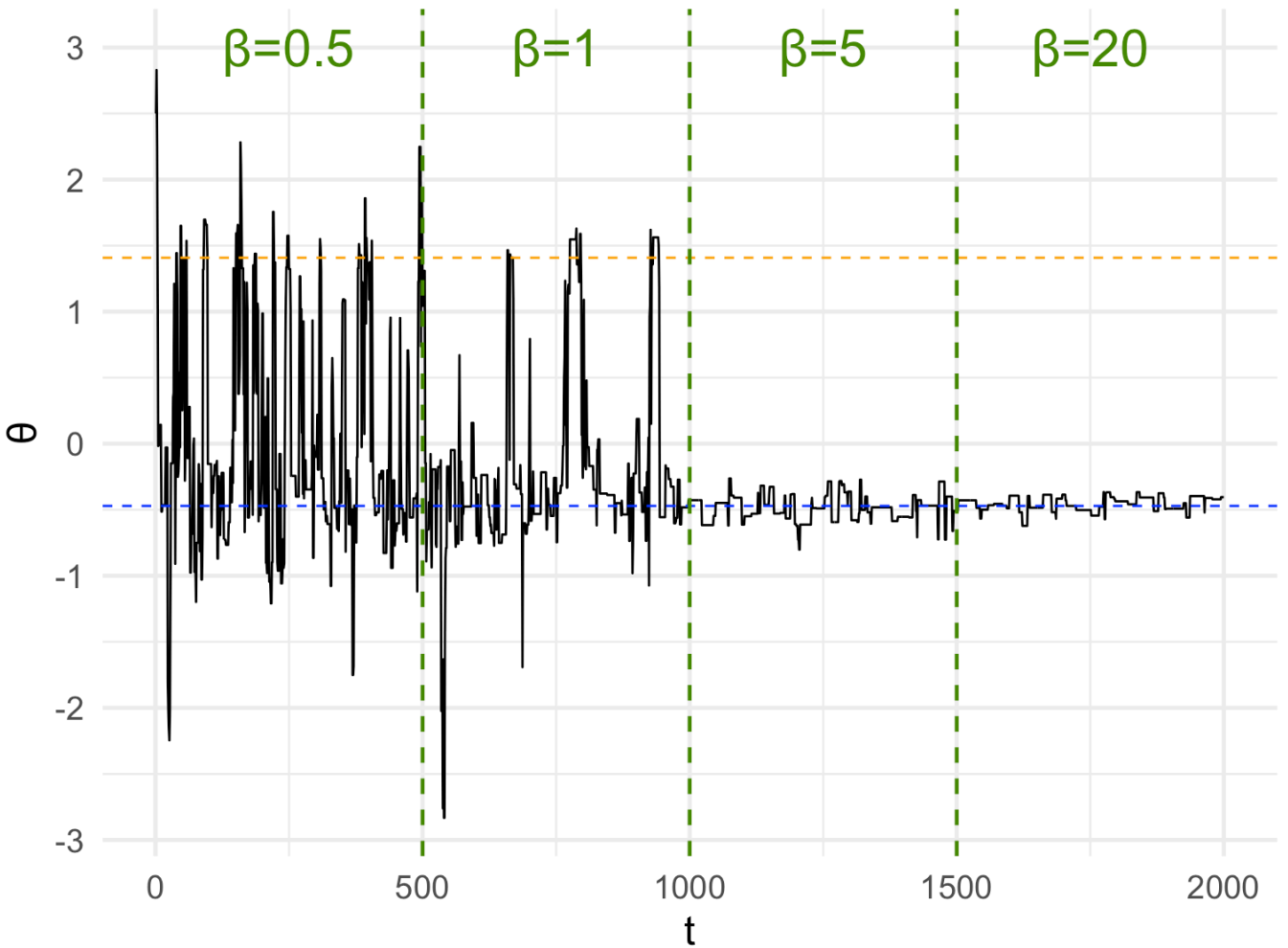
- The inverse temperature β controls the exploration of the space
  - Small β: the chain explores the space widely
  - Large β: the chain concentrates on the global minimum
- Thus, β which was a constant is now β_t
- We will try to increase β along the iterations of the previous algorithm

Simulated annealing algorithm

1. Initialize θ_0
2. Set the cooling schedule: $\beta_1, \beta_2, \ldots, \beta_T$
3. For $t = 1, 2, \ldots, T$:
   - Generate $\zeta \sim g(\zeta|\theta_t)$, a proposal kernel given θ_t
   - Compute the acceptance ratio $\rho = \dfrac{\exp\{-\beta_t h(\zeta)\} g(\theta_t|\zeta)}{\exp\{-\beta_t h(\theta_t)\} g(\zeta|\theta_t)}$
   - Draw $U \sim \text{Unif}(0,1)$ and set $\theta_{t+1} = \zeta \mathbf{1}\{U < \rho\} + \theta_t \mathbf{1}\{U \geq \rho\}$

Nothing changes in the algorithm, except that β is now a function of t.

Piecewise constant cooling schedule



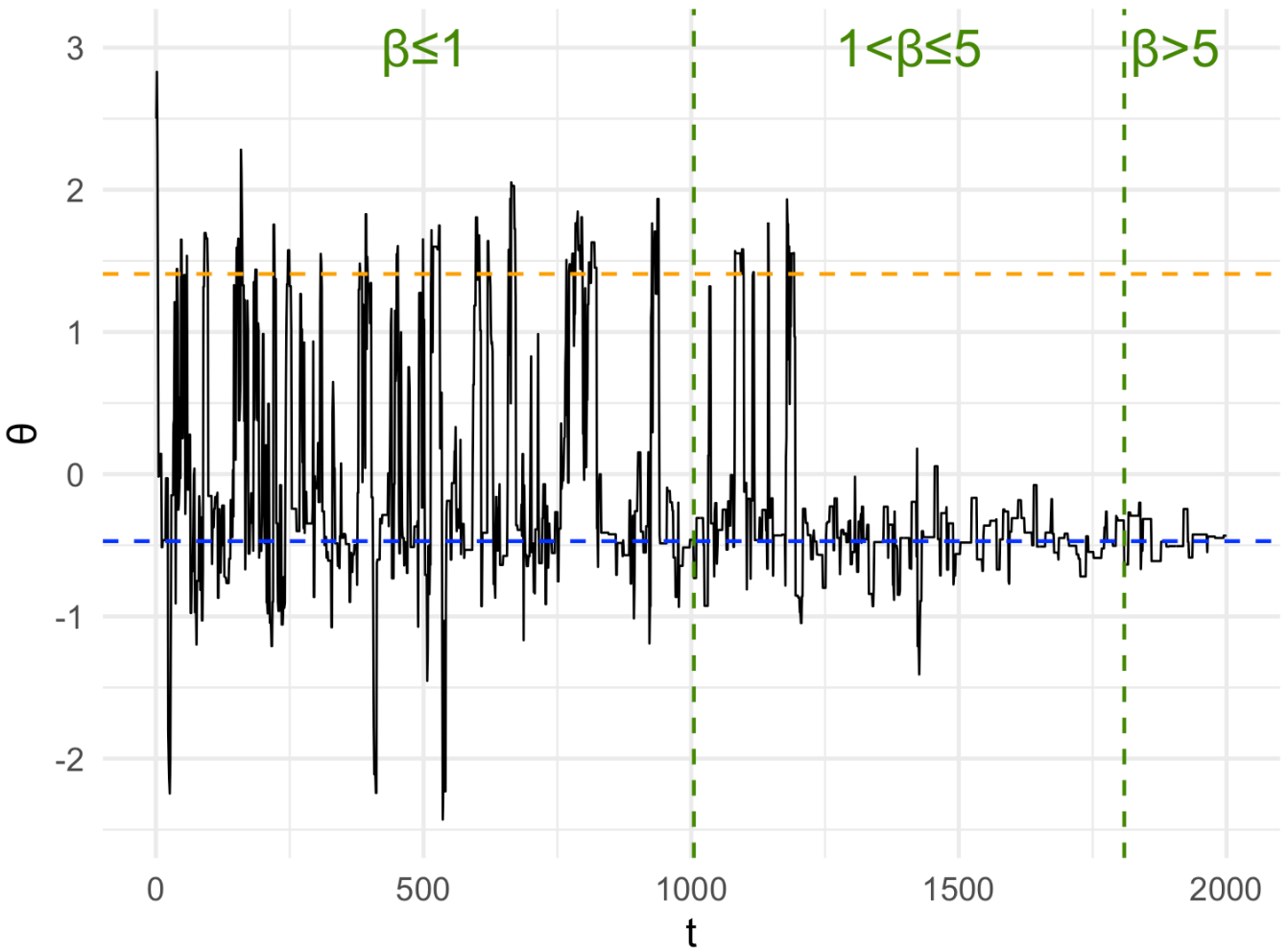
Cooling:

$\beta_t \in \{0.5, 1, 5, 20\}$

Proposal:

$\zeta|\theta_t \sim \mathcal{N}(\theta_t, 1)$

Linear cooling schedule



Cooling:

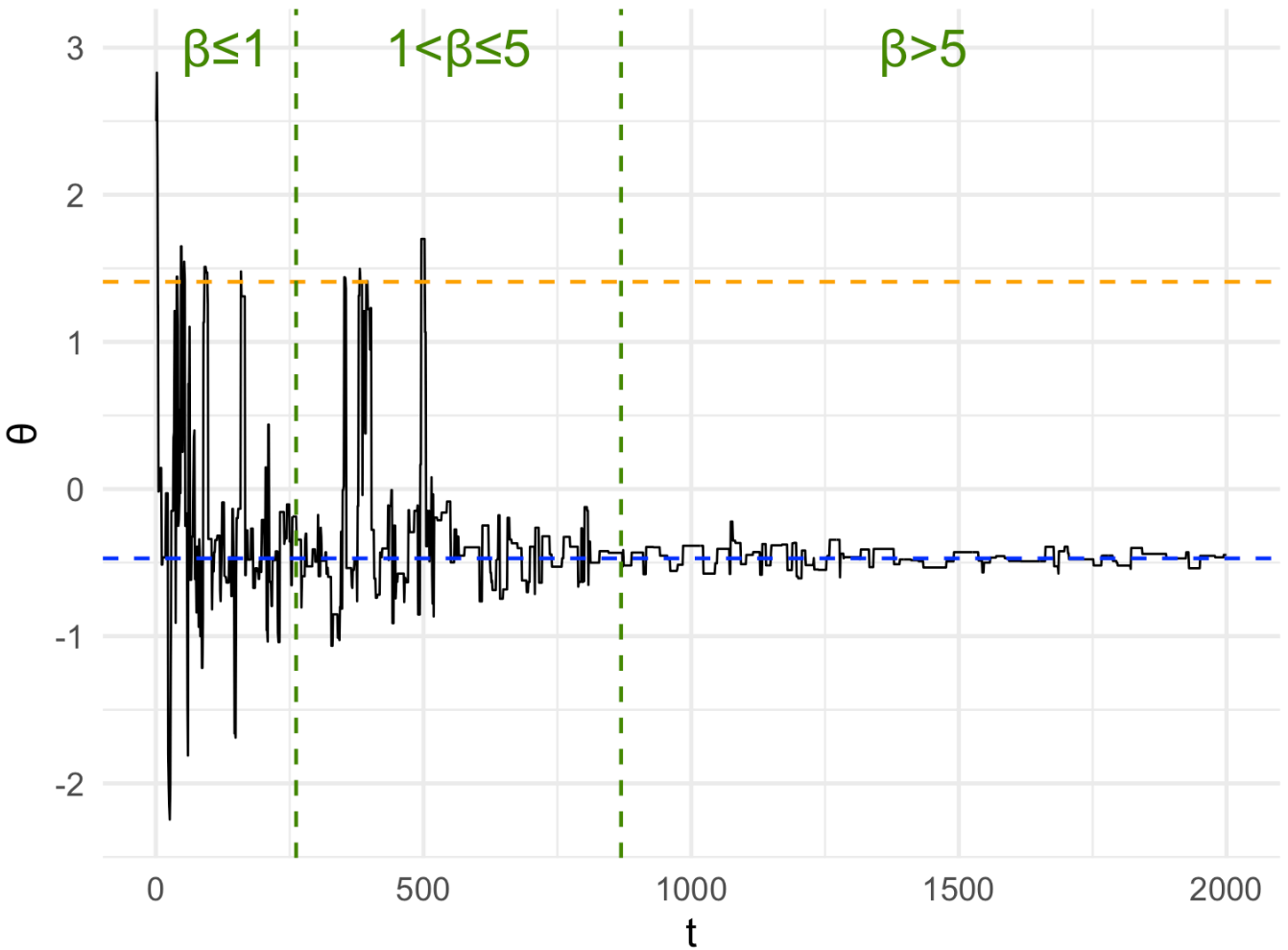
$$\beta_t = 1/(t_0 - \alpha t)$$

with t_0 and α such that $\beta_0 = 0.5$ and $\beta_T = 100$

Proposal:

$$\zeta|\theta_t \sim \mathcal{N}(\theta_t, 1)$$

Exponential cooling schedule



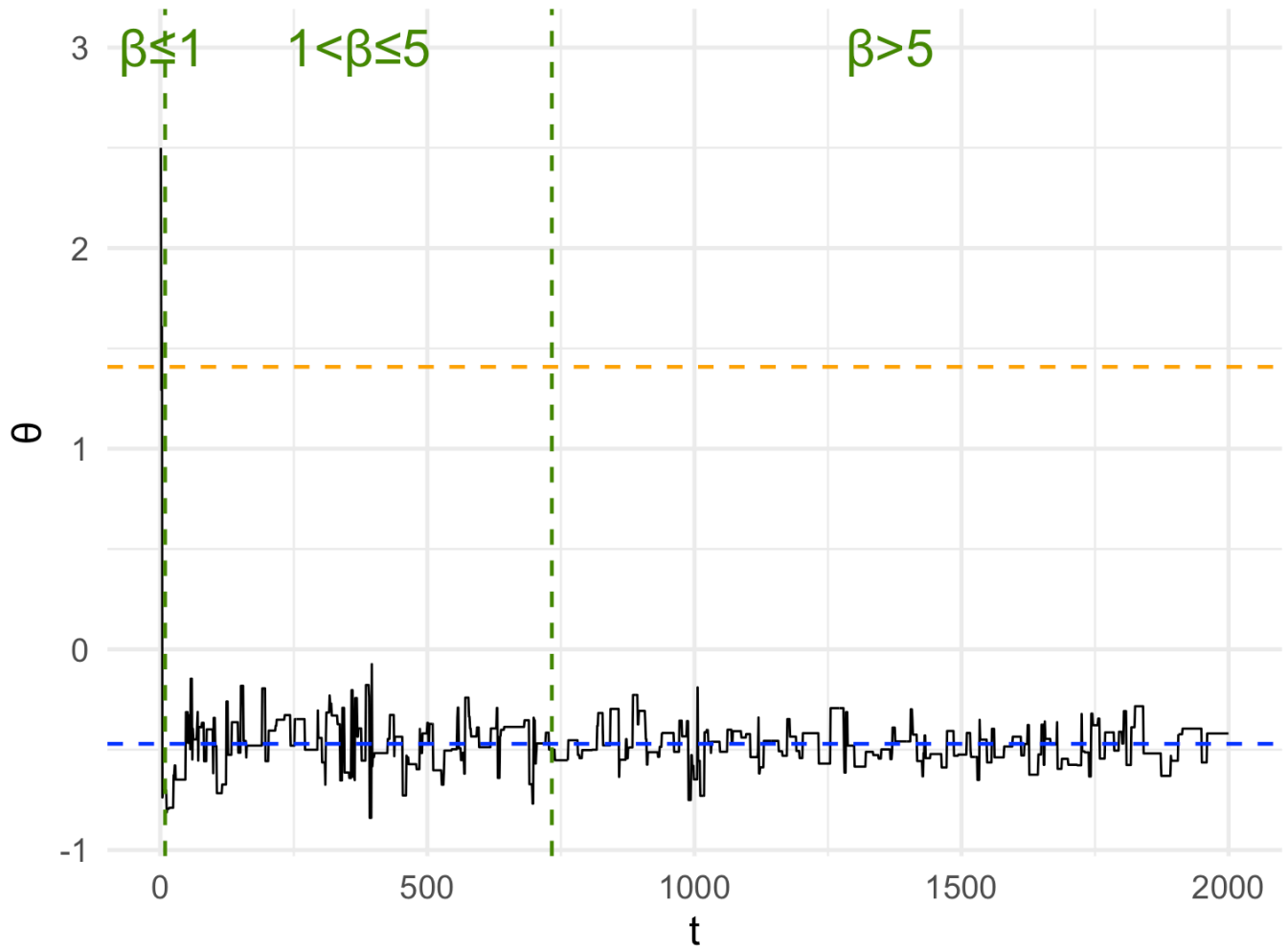
Cooling: $\beta_t = \beta_0 \alpha^t$

with β_0 and α such that $\beta_0 = 0.5$ and $\beta_T = 100$

Proposal:

$\zeta | \theta_t \sim \mathcal{N}(\theta_t, 1)$

Logarithmic cooling schedule



Cooling:

$$\beta_t = \beta_0 \log(1 + kt)$$

with β_0 and k such that $\beta_0 = 0$ and $\beta_T = 6$

Proposal:

$$\zeta|\theta_t \sim \mathcal{N}(\theta_t, 1)$$

Conclusion on the cooling schedules

- **Linear cooling schedule**: $\beta_t = 1/(t_0 - \alpha t)$
  - Spend half of the time exploring the space ($\beta \leq 1$)
  - Approximately 10% of the time around the global minimum ($\beta > 5$)
- **Exponential cooling schedule**: $\beta_t = \beta_0 \alpha^t$
  - Approximately 25% of the time exploring the space ($\beta \leq 1$)
  - Spend half of the time moving around the global minimum ($\beta > 5$)
- **Logarithmic cooling schedule**: $\beta_t = \beta_0 \log(1 + kt)$
  - Spend less that 1% of the time exploring the space ($\beta \leq 1$)
  - Spend more than half of the time moving around the global minimum ($\beta > 5$)

- Sometimes, **non-monotonic** cooling schedules are useful

- The **cooling schedule** is a crucial part of the simulated annealing algorithm
- It is often **problem-dependent**, and should be chosen carefully
  - too slow: may not converge
  - too fast: may not explore the space enough and be stuck at a local minimum
- The **proposal kernel** is also crucial
  - it should be tuned to explore the space efficiently at the beginning
  - thus, calibrated with small values of β

3. Stochastic gradient descent

- **Aim**: $\underset{\theta \in \Theta}{\operatorname{argmin}}\, h(\theta)$ when the function h is **too costly to evaluate** at each iteration
- We consider $h(\theta) = \mathbb{E}(f(\theta, \Xi))$ where $\Xi \sim p$
- and estimates: $\widehat{\nabla h(\theta)} = \partial_\theta f(\theta, \xi)$

Gradient descent algorithm

A recursive algorithm to find the minimum of a function h:

$$\theta_{t+1} = \theta_t - \eta_t \, \nabla h(\theta_t)$$

- The step size η_t may be constant: $\eta_t = \eta$
- It is often chosen as $\eta_t = C/t$
- Stopping rule: $\|\nabla h(\theta_T)\| < \varepsilon$, with a given threshold ε
- May be trapped at a local minimum or at a saddle point

3.1 SGD: the algorithm

- The gradient $\nabla h(\theta_t)$ is replaced by $\partial_\theta f(\theta_t, \xi_t)$, with $\xi_t \sim p$

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta_t \, \partial_\theta f(\hat{\theta}_t, \xi_t), \quad \xi_t \sim p$$

- We have

$$\mathbb{E}\left[\hat{\theta}_{t+1} \middle| \hat{\theta}_t\right] = \hat{\theta}_t - \eta_t \, \mathbb{E}\left[\partial_\theta f(\hat{\theta}_t, \xi_t) \middle| \hat{\theta}_t\right] = \hat{\theta}_t - \eta_t \, \nabla h(\hat{\theta}_t).$$

  Thus,

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta_t \, \nabla h(\hat{\theta}_t) + \eta_t \, \varepsilon_t, \quad \varepsilon_t = \partial_\theta f(\hat{\theta}_t, \xi_t) - \nabla h(\hat{\theta}_t).$$

- $\mathbb{E}(\varepsilon_t) = 0$

- If the step size is held constant, we keep moving around the minimum $\theta^\star$:
  - $\nabla h(\theta^\star) = 0$, i.e. $\mathbb{E}(\partial_\theta f(\theta^\star, \Xi)) = 0$
  - But the noise $\eta\,\varepsilon_t$ has a postive variance
  - Thus, even if $\hat{\theta}_t \approx \theta^\star$, $\hat{\theta}_{t+1} \approx \theta^\star + \eta\,\varepsilon_t \neq \theta^\star$ that does not tend to 0
- Conclusion: the step size should tend to 0 for convergence

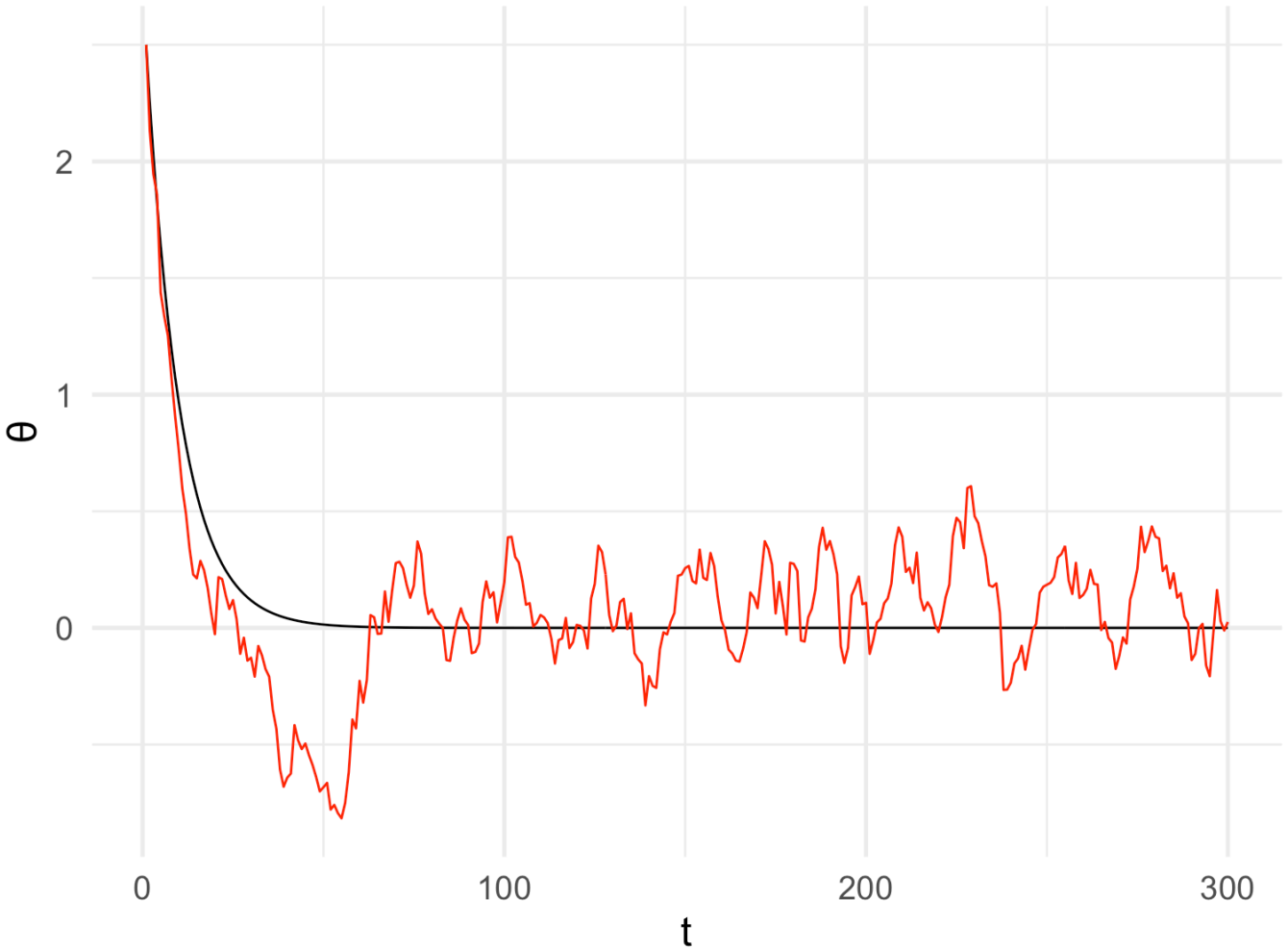
- On the other hand, if $\eta_t \to 0$ too fast,
  - $\hat{\theta}_{t+1} \approx \hat{\theta}_t$ too quickly
  - the algorithm may never see the bottom of the valley

- The step size η_t is chosen so that $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$

3.2 Fake example

- Consider $h(\theta) = \mathbb{E}((\theta - \Xi)^2)$, with $\Xi \sim \mathcal{N}(0, 1)$
- Actually, $h(\theta) = \theta^2 + 1$ and $\theta^\star = 0$
- We estimate the gradient $\nabla h(\theta)$ with $\partial_\theta f(\theta, \xi) = 2(\theta - \xi)$
- GD: $\theta_{t+1} = \theta_t - 2\eta_t \, \theta_t$
- SGD: $\hat{\theta}_{t+1} = \hat{\theta}_t - 2\eta_t \, \hat{\theta}_t + 2\eta_t \xi_t$

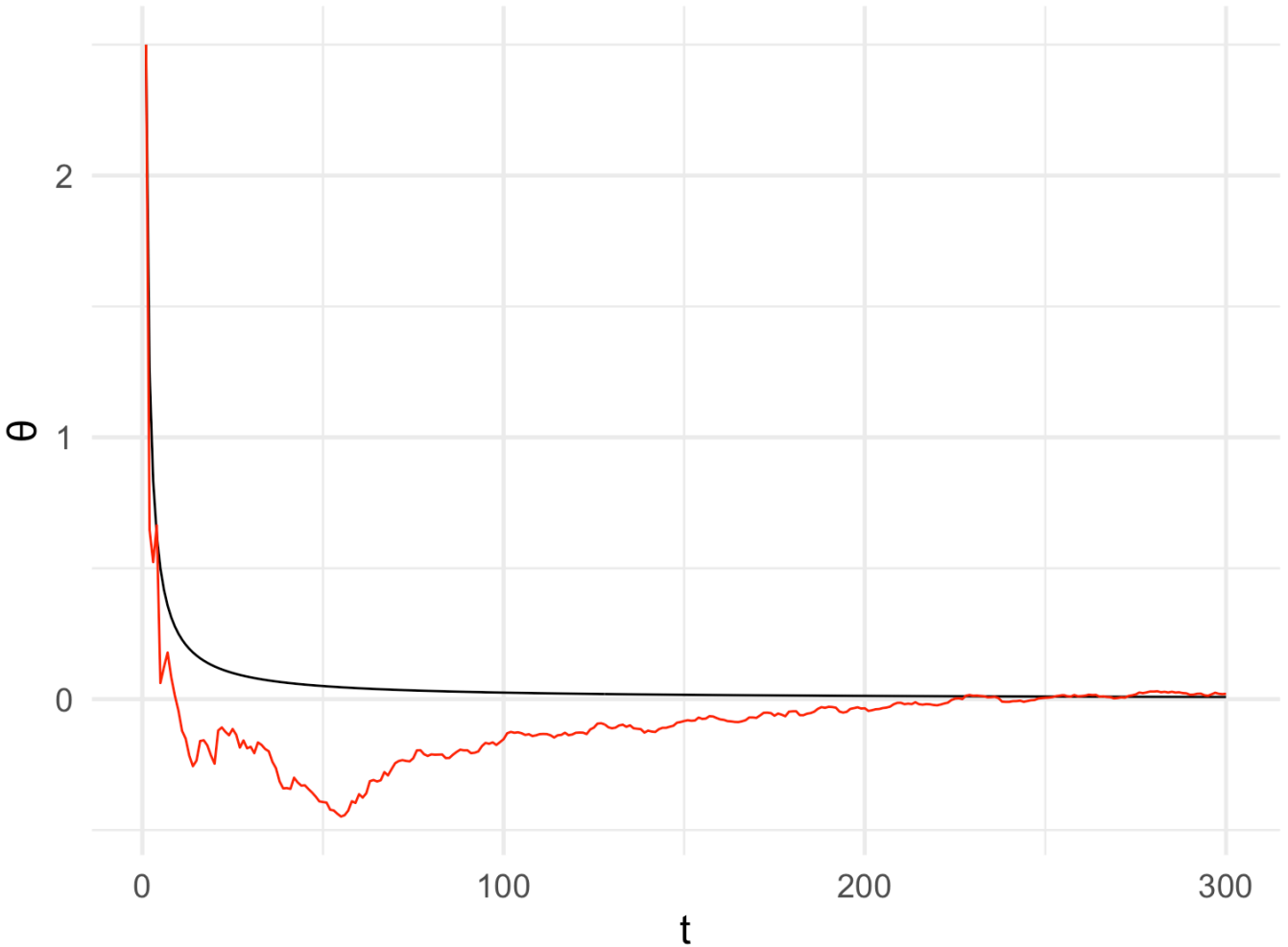
Constant step size $\eta_t = 0.05$



- Black: GD
- Red: SGD

SGD moves around $\theta^\star = 0$ but does not converge

Decreasing step size $\eta_t = 0.5/t$



- Black: GD
- Red: SGD

SGD seems to tend to $\theta^\star = 0$

3.3 Application to machine learning

Quite often in ML, we optimize

$$h(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

where

- n is the number of observations
- $f_i(\theta)$ is the loss function for the i-th observation or $-\log(\text{likelihood})$

Then, for example,

$$h(\theta) = \mathbb{E}\big[f_I(\theta)\big], \quad I \sim \text{Unif}\{1, \ldots, n\}$$

- In this case, the gradient $\nabla h(\theta)$ is estimated by $\partial_\theta f_I(\theta)$
- The error between

$$\nabla h(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\theta) \quad \text{and} \quad \partial_\theta f_I(\theta)$$

  can be large:
  - $\nabla h(\theta)$ is related to all the observations
  - $\partial_\theta f_I(\theta)$ is related to a single observation (drawn at random)
- An idea to reduce the noise is to involve more than one observation at each iteration

Random mini-batches

- To reduce the error, we can use a mini-batch of size $m < n$
- That is to say, we use the fact that

$$h(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta) = \mathbb{E}\left(\frac{1}{m}\sum_{j=1}^{m} f_{I_j}(\theta)\right),$$

  where $\Xi = (I_1, \ldots, I_m)$ is sampled at random without replacement from $\{1, \ldots, n\}$
- The estimated gradient is

$$\frac{1}{m}\sum_{j=1}^{m} \partial_\theta f_{I_j}(\theta)$$

- Since it involves m observations, the variance of the error is reduced

- The SGD algorithm is then

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta_t \, \frac{1}{m} \sum_{j=1}^{m} \partial_\theta f_{I_{t,j}}(\hat{\theta}_t)$$

  where $\Xi_t = (I_{t,1}, \ldots, I_{t,m})$ is sampled at random without replacement from $\{1, \ldots, n\}$

- m is the **mini-batch size**
- η_t is the step size or the **learning rate**

A basic numerical example

- Consider a large dataset of size $n = 10^6$: $y_1, \ldots, y_n$
- We want to fit a model that predicts y with a constant θ
- The loss function is the squared error: $f_i(\theta) = (\theta - y_i)^2$
- The function to minimize is

$$h(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = \frac{1}{n} \sum_{i=1}^{n} (\theta - y_i)^2$$

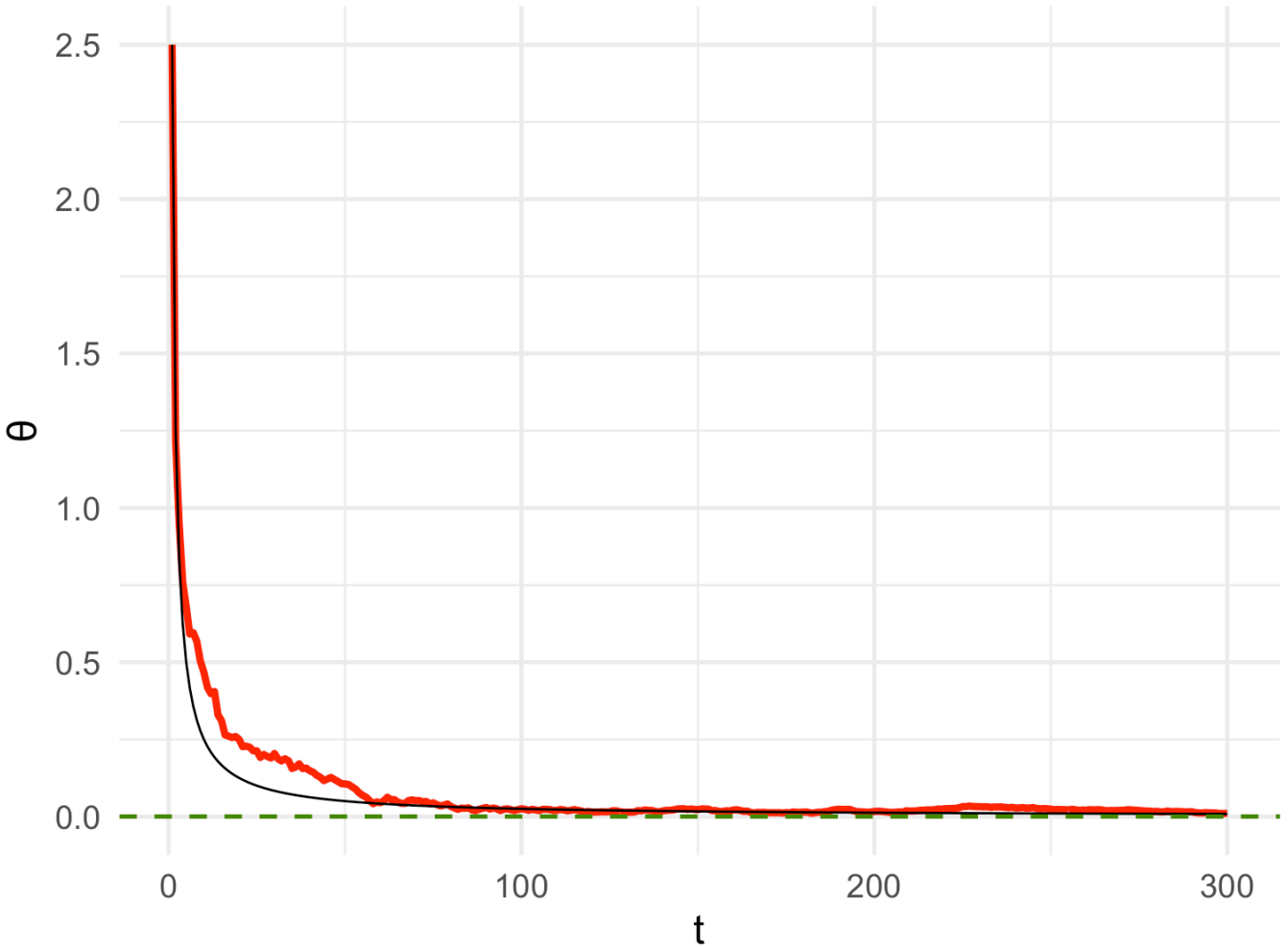
- $\partial_\theta f_i(\theta) = 2(\theta - y_i)$

Mini-batch of size $m = 1$

- Black: GD
- Red: SGD

$$\eta_t = 0.5/t$$

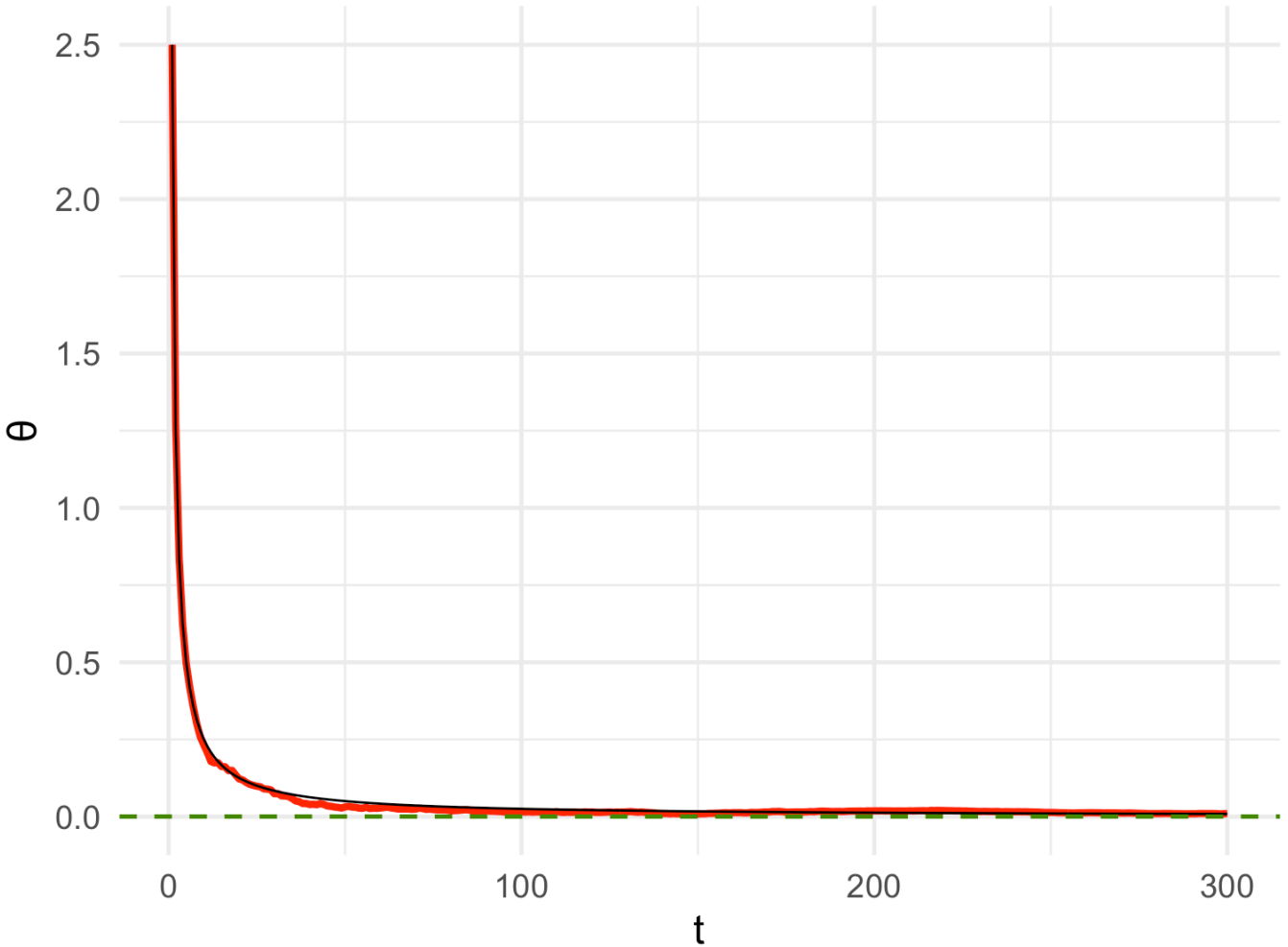
Mini-batch of size $m = 10$



- Black: GD
- Red: SGD

$$\eta_t = 0.5/t$$

Mini-batch of size $m = 100$



- Black: GD
- Red: SGD

$$\eta_t = 0.5/t$$

4. The Expectation-Maximization (EM) algorithm

See second year of the Master's degree