ASTR8150/PHYS8150
Error estimation & Monte Carlo methods

Fabien Baron

Georgia State University

fbaron@gsu.edu

Fall 2025

Monte Carlo methods

- Monte Carlo methods are **stochastic** methods that make use of repeated random or **pseudorandom** sampling to often (but not always) tackle problems too hard to solve analytically/numerically
- Named after the Monte Carlo casino (Monaco)
- Three main classes of applications: **probability distribution sampling, optimization, numerical integration**

Monte Carlo methods (prob. sampling, opt., num. int.)

- Determine the statistical properties of possible inputs
- Generate many sets of possible inputs which follows the above properties
- Perform a deterministic calculation with these sets
- Analyze statistically the results
- The strength of Monte Carlo methods is that the error on the results typically decreases as $1/\sqrt{N}$

Probability sampling: Monte Carlo simulations

- MC simulations/methods allow estimation of both **random** and **systematic** errors
- Common procedure to all MC methods:
  - Determine the probability distribution of inputs (priors!)
  - Generate $N \gg 1$ sets of possible inputs which follows this distribution
  - Transform the input into desired output (a deterministic calculation)
  - Examine the probability distribution of results ; typical error on these results generally decreases as $1/\sqrt{N}$
- Random errors can be estimated directly from the output distribution
- Systematic errors can be estimated by feeding known input values to the MC simulation, then check them against the final mean parameters

Monte Carlo simulations: examples

- Implement Monte Carlo simulation of these:
  - $\boldsymbol{X} \sim \mathcal{N}(0,1)$ and $\boldsymbol{Y} \sim \mathcal{N}(0,1)$, what is the distribution of $\boldsymbol{Z} = \boldsymbol{X} + \boldsymbol{Y}$, $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Y}$, $\boldsymbol{Z} = \boldsymbol{X}/\boldsymbol{Y}$?
  - What is the probability to obtain either 3, 6 or 9 heads if one draws a coin ten times?
  - Only 1% of the population who participate to a zombie plague screening have the plague. 95% of plague carriers will get tested positive, and 10% of non-carriers also get tested positive. Fabien just got tested positive. What is the probability he is infected?

Monte Carlo error estimation: bootstraping

- A **resampling method** using MC simulation: uses the data you already have collected to analyze the effect of uncertainty in your measurements: seems akin to "pull oneself up by one's bootstraps"
- The underlying idea: considering your full data set is M data points, the probability of a set which includes only the first $M-1$ plus a repeat of one of these $M-1$ ones is high.
- Bootstrap = resample the data with replacement, keeping the size of the resample equal to the size of the original data set. A **bootstrap data set** is made by picking M points out of M with possible redundancy (aka **with replacement**) and/or missing points is a probable data set.
- Repeatedly generate $N \gg 1$ such data sets
- For each of these, transform the data inputs into the desired output
- Characterize the distribution resulting from the N samples
- They are several bootstrap resampling techniques, some more advanced than others (e.g. wild bootstrap for heteroskedastic data).

Monte Carlo error estimation: the jackknife

- The jackknife is another resampling method that predates the bootstraping methods.
- The jackknife estimator of a parameter is found by systematically leaving out each observation from a dataset and calculating the estimate and then finding the average of these calculations: given a sample of size N, the jackknife estimate is found by averaging the estimates of each $N - 1$-sized sub-sample.

Monte Carlo error estimation: bootstrapping model-fitting

- Bootstrapping to estimate errors on model parameters when doing model fitting:
  - For each data set built from the original data by boostrapping, apply your model-fitting procedure.
  - Store the parameters obtained by model fitting
  - Plot the Monte Carlo histogram for each parameter
  - Derive error bars based on the shape of the histogram

Monte Carlo error estimation: bootstrapping line fit

- Example: Let's first generate N data triplets (x_i, y_i, σ_i), $i = 1 \dots N$. At times $\boldsymbol{X} \sim U(0,1)$ we measured $\boldsymbol{Y} = \theta_1 + \theta_2 \boldsymbol{X} + \boldsymbol{n}$ where $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{\sigma}^2)$ with $\boldsymbol{\sigma} \sim U(1,2)$. We can pick $\theta_1 = 5$ and $\theta_2 = 7$ to create the data set.
- Now we want to estimate the most probable (θ_1, θ_2). Express the χ^2. Solve for θ_1 and θ_2 using any solver (e.g. Nelder-Mead, BFGS, Levenberg).
- Now we want to estimate the error bars using boostrap. For this we're going to generate bootstraped data sets, made from randomly picking N (x_i, y_i, σ_i) triplets from the original data. There will probably be missing points and redundant points in these new data sets. Generate 10000 of these datasets, and for each of these estimate (θ_1, θ_2).
- How does the result vary with the number of data and noise level? With the number of MC bootstraps? How does this compare to uncertainties you may expect from the χ^2 distribution? Try to add systematic errors to the noise.

Monte Carlo error estimation: bootstrapping examples

- Another classic example: the correlation coefficient of $\boldsymbol{X}$ with $\boldsymbol{Y}$ is given by $\rho = \frac{<(x-\mu_x)(y-\mu_y)>}{\sigma_x \sigma_y} = \frac{<xy>-\mu_x\mu_y}{\sigma_x\sigma_y}$, which possible range is $-1 \leq \rho \leq 1$ (more details on this when we do time series)
- Trick to generate Gaussian correlated data: if $\boldsymbol{X} \sim \mathcal{N}(0,1)$ and $\boldsymbol{Y} \sim \mathcal{N}(0,1)$ are uncorrelated, then $\boldsymbol{Z} = \rho \boldsymbol{X} + \sqrt{1-\rho^2}\boldsymbol{Y}$ and $\boldsymbol{X}$ are correlated with correlation factor ρ. Also by definition, $\boldsymbol{X'} = \mu_{X'} + \sigma_{X'}\boldsymbol{X}$ and $\boldsymbol{Z'} = \mu_{Z'} + \sigma_{Z'}\boldsymbol{Z}$ will stay correlated with the same ρ.
- Exercise: generate two correlated vectors of 100 elements with $\rho = 0.7$. This constitutes your data. Now compute error bars on the correlation coefficient through bootstrap.

Monte Carlo error estimation: residual bootstrap

Another approach to bootstrapping in regression problems is to resample residuals:

- Fit the model and retain the fitted values $\hat{y}_i$ and the residuals $\hat{\epsilon}_i = y_i - \hat{y}_i, (i = 1, \ldots, n)$. Then for each data pair, (x_i, y_i) add a randomly resampled residual $\hat{\epsilon}_j$ to the response variable y_i, to create a new set $y_i^* = y_i + \hat{\epsilon}_j$ where j is selected randomly from the list $(1, \ldots, n)$ for every i.
- Refit the model using the fictitious data y_i^*, and retain the fitted parameters estimated from the synthetic y_i^*. Repeat the previous steps a large number of times and study the distribution of the parameters.
- Residuals to resample can be the raw residuals or studentized residuals (in linear regression).

Monte Carlo error estimation: wild boostrap

- The wild bootstrap is for heteroskedastic data, for which classic and residual bootstrap can fail.
- The wild bootstrap also resamples the response variable based on the residuals values: $y_i^* = \hat{y}_i + \hat{\epsilon}_i v_i$ so the residuals are randomly multiplied by a random variable v_i with mean 0 and variance 1. This method assumes that the 'true' residual distribution is symmetric and can offer advantages over simple residual sampling for smaller sample sizes. The random variable v_i can be chosen to follow a standard normal distribution, the Rademacher distribution, the Mammen distribution (1993):

$$v_i = \begin{cases} -(\sqrt{5}-1)/2 & \text{with prob. } (\sqrt{5}+1)/(2\sqrt{5}) \\ (\sqrt{5}+1)/2 & \text{with prob. } (\sqrt{5}-1)/(2\sqrt{5}) \end{cases} \tag{1}$$

Error analysis using Fisher information

- What is the sensitivity of the likelihood $f(\theta)$ to $\theta_1, \theta_2, \ldots$?
- Or rather, the sensitivity of the loglikelihood $\log f(\theta)$?
- Its gradient (also called the score) is $\frac{\partial \log f}{\partial \theta}$. It measures in which direction (sign of the gradient) the likelihood changes when θ does, and how strongly (amplitude of the gradient).
- Its second derivative is called the Hessian, $\frac{\partial^2 \log f}{\partial \theta^2}$. It measures the curvature of the likelihood with respect to the parameters; so how influenced the likelihood is by variations of θ.
- The Fisher information $I(\theta)$ measures how much information an observable random variable X carries about θ. Two ways of calculating it:
  - As the square of the expected value of the gradient:
    $I(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial \theta} \log f(X;\theta)\right)^2\right]$
  - As the negative expected value of the second derivative:
    $I(\theta) = -\mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log f(X;\theta)\right]$

Cramér-Rao Lower Bound:

- Large Fisher information $\rightarrow$ small variance of the likelihood $\rightarrow$ more accurate estimation.
- Small Fisher information $\rightarrow$ less precise estimation.
- No unbiased estimator of θ can have a variance smaller than the reciprocal of the Fisher information. This means there is a lower bound to the uncertainty one can get on θ, the Cramér-Rao Lower Bound: $\mathrm{var}(\hat{\theta}) \geq \frac{1}{I(\theta)}$
- The maximum likelihood estimator is an unbiased estimator (= setting the gradient of the loglikehood to zero and solving, when possible).
- So uncertainties for maximum likelihood will be exactly: $\mathrm{std}(\hat{\theta}) = \sqrt{\frac{1}{I(\theta)}}$

Fisher information for repeat measurements of a distribution

- We have $X \sim \mathcal{N}(\mu, \sigma^2)$ and measure $x_1, \ldots, x_N$
- Likelihood function: $L(\mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$
- Log-likelihood: $\log L(\mu) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$
- First derivative: $\frac{\partial}{\partial\mu}\log L(\mu) = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu)$
- To compute the Fisher information, we can take the expectation of the square of the derivative:
$$I_N(\mu) = \mathbb{E}\left[\left(\frac{\partial}{\partial\mu}\log L(\mu)\right)^2\right] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma^2}\right)^2\right] = \frac{1}{\sigma^2}\cdot\mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^2\right] = \frac{N}{\sigma^2}$$
- Or directly with the second derivative:
$$I_N(\mu) = -\mathbb{E}\left[\frac{\partial^2}{\partial\mu^2}\log L(\mu)\right] = -\mathbb{E}\left[-\frac{N}{\sigma^2}\right] = \frac{N}{\sigma^2}$$

Fisher information for linear measurement problem

- We consider the linear model: $\mathbf{y} = A\mathbf{x} + \mathbf{n}$
  - $\mathbf{y} \in \mathbb{R}^m$ is the observed data,
  - $A \in \mathbb{R}^{m \times n}$ is a known matrix,
  - $\mathbf{x} \in \mathbb{R}^n$ is the unknown parameter vector,
  - $\mathbf{n} \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise with known covariance $\Sigma \succ 0$.

  Since $\mathbf{n} \sim \mathcal{N}(0, \Sigma)$, it follows that: $\mathbf{y} \sim \mathcal{N}(A\mathbf{x}, \Sigma)$
- Log-Likelihood: $\log p(\mathbf{y}; \mathbf{x}) \sim -\frac{1}{2} \log |\Sigma| - \frac{1}{2}(\mathbf{y} - A\mathbf{x})^\top \Sigma^{-1}(\mathbf{y} - A\mathbf{x})$
- Gradient: $\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x}) = A^\top \Sigma^{-1}(\mathbf{y} - A\mathbf{x})$
- Fisher with first derivative:

$$\begin{aligned}
\mathcal{I}(\mathbf{x}) &= \mathbb{E}\left[\left(\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right) \left(\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right)^\top \right] \\
&= A^\top \Sigma^{-1} \, \mathbb{E}\left[(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^\top \right] \Sigma^{-1} A \\
&= A^\top \Sigma^{-1} \Sigma \Sigma^{-1} A = A^\top \Sigma^{-1} A \text{ , since } \mathbf{y} - A\mathbf{x} = \mathbf{n} \sim \mathcal{N}(0, \Sigma)
\end{aligned}$$

- Fisher with second derivative:
  $\mathcal{I}(\mathbf{x}) = -\mathbb{E}\left[\nabla^2_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right] = -\left(-A^\top \Sigma^{-1} A\right) = A^\top \Sigma^{-1} A$

Fisher information for linear measurement problem

- We consider the linear model: $\mathbf{y} = A\mathbf{x} + \mathbf{n}$
  - $\mathbf{y} \in \mathbb{R}^m$ is the observed data,
  - $A \in \mathbb{R}^{m \times n}$ is a known matrix,
  - $\mathbf{x} \in \mathbb{R}^n$ is the unknown parameter vector,
  - $\mathbf{n} \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise with known covariance $\Sigma \succ 0$.

  Since $\mathbf{n} \sim \mathcal{N}(0, \Sigma)$, it follows that: $\mathbf{y} \sim \mathcal{N}(A\mathbf{x}, \Sigma)$
- Log-Likelihood: $\log p(\mathbf{y}; \mathbf{x}) \sim -\frac{1}{2} \log |\Sigma| - \frac{1}{2}(\mathbf{y} - A\mathbf{x})^\top \Sigma^{-1}(\mathbf{y} - A\mathbf{x})$
- Gradient: $\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x}) = A^\top \Sigma^{-1}(\mathbf{y} - A\mathbf{x})$
- Fisher with first derivative:

$$\begin{aligned}
\mathcal{I}(\mathbf{x}) &= \mathbb{E}\left[\left(\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right) \left(\nabla_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right)^\top \right] \\
&= A^\top \Sigma^{-1} \, \mathbb{E}\left[(\mathbf{y} - A\mathbf{x})(\mathbf{y} - A\mathbf{x})^\top \right] \Sigma^{-1} A \\
&= A^\top \Sigma^{-1} \Sigma \Sigma^{-1} A = A^\top \Sigma^{-1} A \text{ , since } \mathbf{y} - A\mathbf{x} = \mathbf{n} \sim \mathcal{N}(0, \Sigma)
\end{aligned}$$

- Fisher with second derivative:
  $\mathcal{I}(\mathbf{x}) = -\mathbb{E}\left[\nabla^2_{\mathbf{x}} \log p(\mathbf{y}; \mathbf{x})\right] = -\left(-A^\top \Sigma^{-1} A\right) = A^\top \Sigma^{-1} A$

Fisher information for the linear model

- Model:

$$y = \theta_1 x + \theta_2 + n, \quad n \sim \mathcal{N}(0, \Sigma)$$

- Vector Form:

$$y \in \mathbb{R}^n, \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$y \sim \mathcal{N}(X\theta, \Sigma)$$

- Log-Likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{2}(y - X\theta)^\top \Sigma^{-1}(y - X\theta) + \text{const}$$

- Gradient:

$$\nabla_\theta \mathcal{L}(\theta) = X^\top \Sigma^{-1}(y - X\theta)$$

- Fisher Information:

$$\mathcal{I}(\theta) = \mathbb{E}[-\nabla_\theta^2 \mathcal{L}(\theta)] = X^\top \Sigma^{-1} X$$

MCMC Optimization: local minima, global minima and test functions

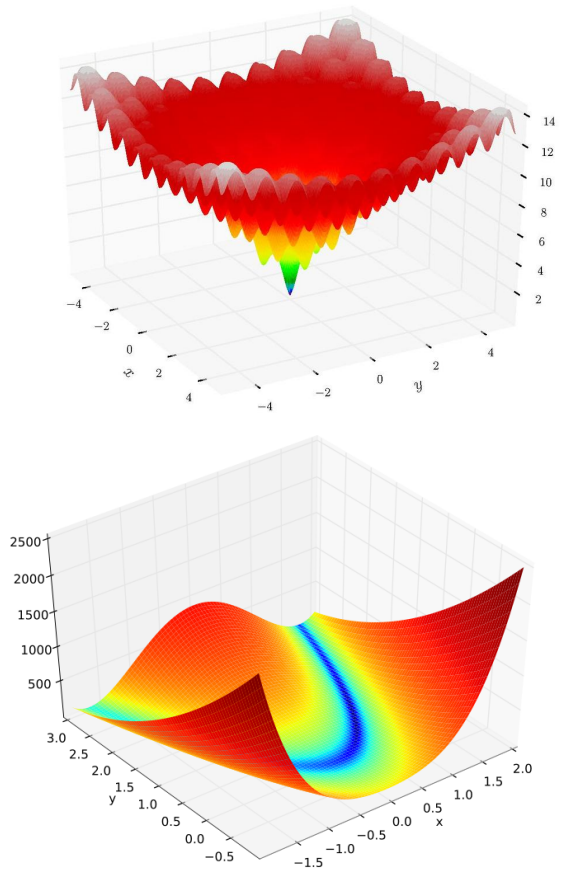


Figure: Ackley (top) and Rosenbrock (bottom) functions in 2D

- Ackley's function in two dimensions: for $-5 \leq x, y \leq 5$

$$f(x,y) = -20\exp\left(-0.2\sqrt{0.5\left(x^2+y^2\right)}\right) - \exp\left(0.5\left[\cos\left(2\pi x\right)+\cos\left(2\pi y\right)\right]\right) + e + 20$$

  has many **local minima** but also a **global minima** $f(0,0) = 0$.

- Rosenbrock function $f(x,y) = (a-x)^2 + b(y-x^2)^2$ has a global minimum at $f(a,a^2) = 0$. Usual values $a = 1$ and $b = 100$.

MCMC Optimization: basic Hill climbing

- Hill climbing constitutes the most basic optimizer for model fitting: we want to maximize a function $\Pr(\boldsymbol{\theta}|\boldsymbol{D})$.
- θ^i represents the value of θ at the ith iteration

Algorithm 1 Hill climbing

procedure HILLCLIMB($f, \boldsymbol{\theta}$) ▷ climb up f
  $\theta^0 \in \pi(\theta)$ ▷ Initialize with values drawn from prior
  for $i = 1, \ldots$ **do**
    $\theta^{\text{trial}} \sim q(\theta^{\text{trial}}|\theta^{i-1})$ ▷ Trial state from proposal distribution q
    if $\Pr(\theta^{\text{trial}}|D) > \Pr(\theta^{i-1}|D)$ **then**
      $\theta^i \leftarrow \theta^{\text{trial}}$ ▷ Accept the move
    else
      $\theta^i \leftarrow \theta^{i-1}$ ▷ State is unchanged
    end if
  end for
end procedure

MCMC Optimization: trial "moves"

- This is the part of the procedure which is stochastic
- Initialization: ideally we start from a random point every time
- How do we pick θ^{trial}?
- We **sample from the prior** = we draw θ^{trial} following its probability distribution, discarding zero prior probability choices
- A classic strategy is to change $\theta^{\text{trial}} = \theta_{i-1} + \delta$ where δ is a **step**
- δ can be a function of the current posterior (better posterior implies smaller δ) and/or iteration number (high iteration number implies smaller δ)
- In practice, codes generally minimize negative loglikelihoods $-\log \Pr(\boldsymbol{\theta}|\boldsymbol{D})$, for example χ^2, rather than directly work maximize $\Pr(\boldsymbol{\theta}|\boldsymbol{D})$.

Markov Chain: definition

- The **Markov property** for a time-dependent process is "the past is conditionally independent of the future given the present state of the process" or "given the present state, the past contains no additional information on the future evolution of the system", or "the future state only depends on the present state".
- A discrete-time process $\theta^t, \theta^{t+1}, \theta^{t+2}, \ldots$ (indexed here by t) is a **Markov Chain** (of order 1) if it has the Markov property: $\Pr(\theta^{t+1}|\theta^1, \theta^2, \ldots, \theta^t) = \Pr(\theta^{t+1}|\theta^t)$

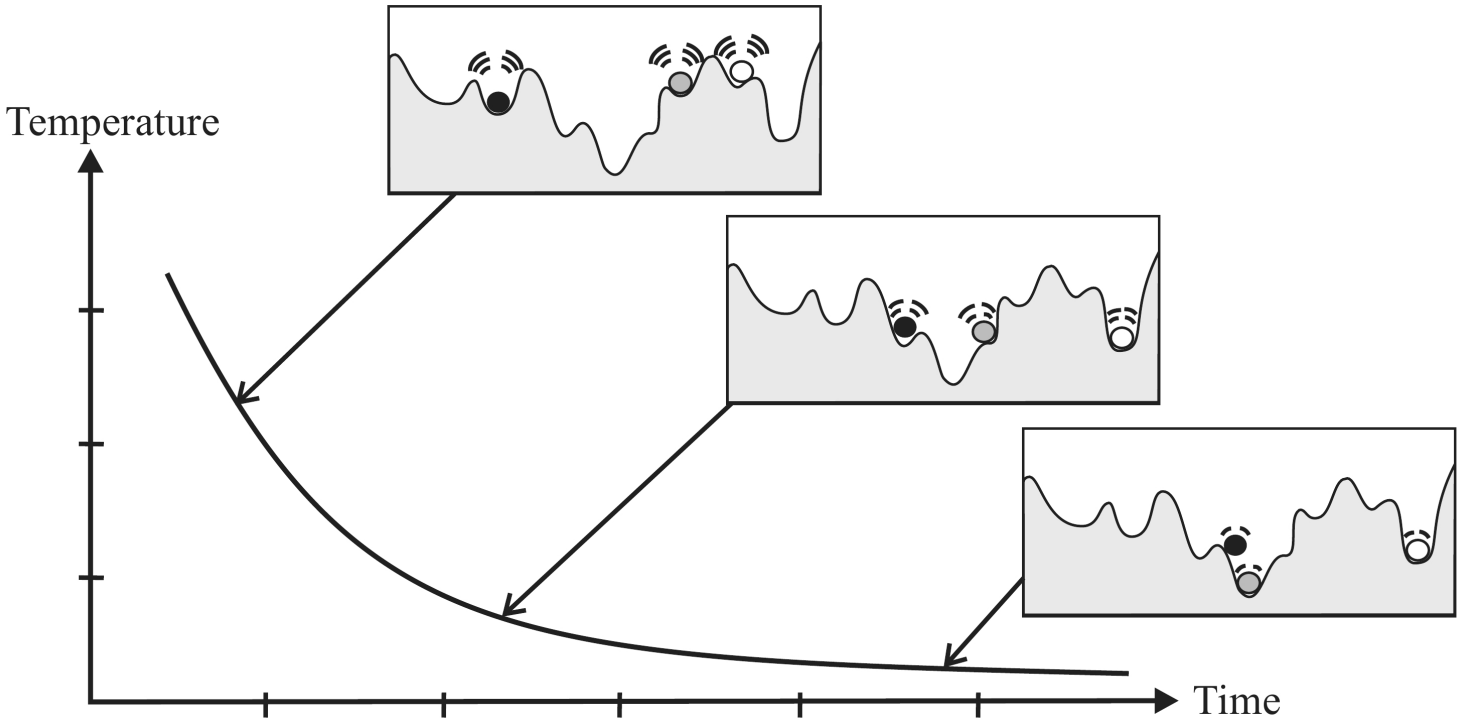
Markov Chain: properties

- A Markov chain Monte Carlo (MCMC) is said time-**homogeneous** if $\Pr(\theta^{t+1} = j|\theta^t = i) = \Pr(\theta^1 = j|\theta^0 = i)$
- For a time-homogeneous Markov Chain, $p_{ij} = \Pr(\theta^{t+1} = j|\theta^t = i)$ does not change with t.
- A Markov Chain is called an **ergodic or irreducible** chain if it is possible to go from every state to every state (not necessarily in one move).
- A Markov Chain is **reversible** if it follows the **detailed balance equation** there is a probability distribution π so that

$$\pi_i \Pr(\theta^{t+1} = j|\theta^t = i) = \pi_j \Pr(\theta^{t+1} = i|\theta^t = j) \tag{2}$$

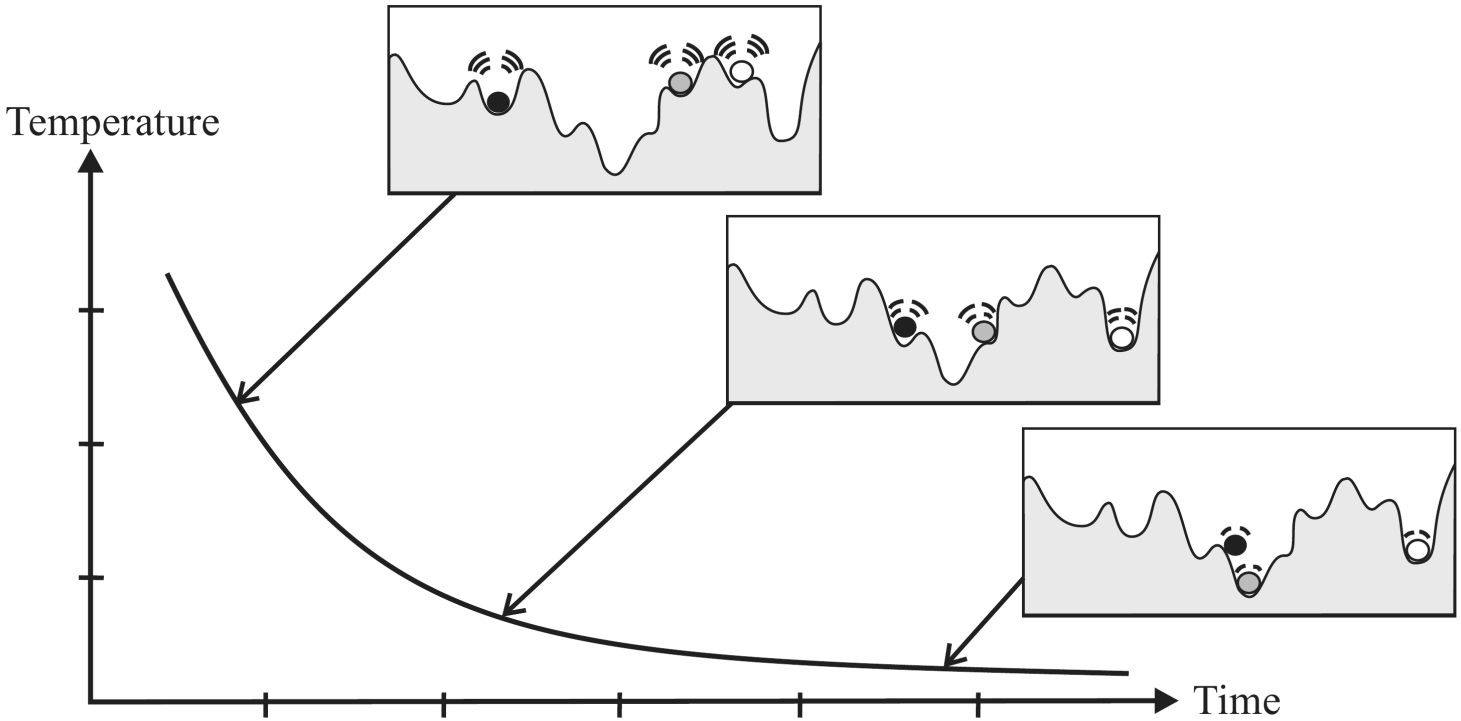
- For a reversible homogenous Markov Chain: $\pi_i p_{ij} = \pi_j p_{ji}$
- A Markov Chain is uniquely defined by its transition probabilities

MCMC Optimization: simulated annealing



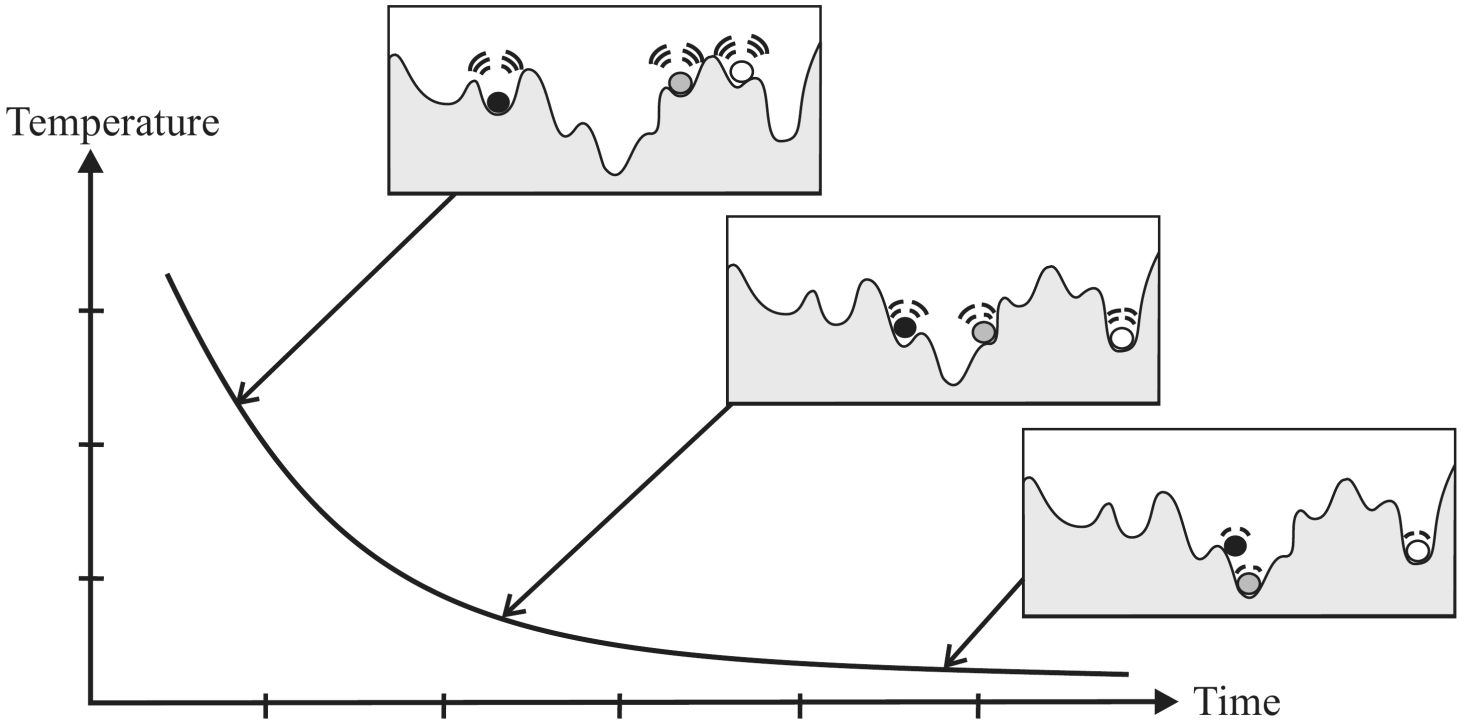
- Anneal: heat (metal or glass) and allow it to cool slowly, in order to remove internal stresses and toughen it.
- Simulated annealing is a **metaheuristic**, i.e. a procedure to search for solutions given limited information about the problem.
- Uses a **temperature schedule** governing acceptance of non-optimal moves with non-zero probabilities to find the global optimum.

MCMC Optimization: simulated annealing



- At hot temperatures, the model parameters are hot, i.e. fairly free to change, they adopt any new values, even possibly ones with worse posterior. Hot temperatures allow exploration of the posterior landscape. This phase is called the **burn-in**.
- As temperature cools down, the model parameters change more slowly, they do not explore as much and tend to stay more localized.

MCMC Optimization: convergence



- As temperature gets cold, the model parameters should settle progressively and may even get "frozen". Convergence tests can decide whether the chain is cold enough to stop optimization.
- The sampling of the parameters outside of the burn-in phase should probe their distributions. These are characterized (means, variances, etc.) to find the MCMC "solution".

MCMC Optimization: Metropolis-Hastings transitions

Algorithm 2 Metropolis-Hastings algorithm

procedure $\mathrm{MH}(f, \boldsymbol{\theta})$ ▷ Metropolis-Hastings
  $\theta^0 \in \pi(\theta)$ ▷ Initialize with values drawn from prior
  for $i = 1, \ldots$ **do**
    $\theta^{\text{trial}} \sim q(\theta^{\text{trial}} | \theta^{i-1})$ ▷ Trial state from proposal distribution q
    $a(\theta^{\text{trial}} | \theta^{i-1}) = min\left\{1, \frac{q(\theta^{i-1}|\theta^{\text{trial}})\,\Pr(\theta^{\text{trial}}|D)}{q(\theta^{\text{trial}}|\theta^{i-1})\,\Pr(\theta^{i-1}|D)}\right\}$ ▷ Acceptance probability
    if $U(0,1) < a(\theta^{\text{trial}} | \theta^{i-1})$ **then**
      $\theta^i \leftarrow \theta^{\text{trial}}$ ▷ Accept the move
    else
      $\theta^i \leftarrow \theta^{i-1}$ ▷ State is unchanged
    end if
  end for
end procedure

MCMC Optimization: simulated annealing

- Incorporate the temperature schedule $0 < t_1 < \ldots < t_i < \ldots < t_N = 1$, for iterations $i = 1 \ldots$ in the simulated annealing minimization
- We want to maximize $\Pr(\theta|D)$, and we do it by maximizing $\Pr(\theta|D)^{\frac{1}{t_i}}$ as iterations and t_i decrease.
- At high temperatures, the algorithm explores the parameter space; at low temperatures, it restricts its exploration
- Step size for the random moves is often tied to the temperature, and reversible: $q(\theta^{\text{trial}}|\theta^{i-1}) = q(\theta^{i-1}|\theta^{\text{trial}})$
- We can define energy as $E(\theta) = -\log \Pr(\theta|D)$
- Simulated annealing is Metropolis-Hastings with an acceptance probability of:

$$a(\theta^{\text{trial}}|\theta^{i-1}) = \min\left\{1, \frac{q(\theta^{i-1}|\theta^{\text{trial}})\Pr(\theta^{\text{trial}}|D)}{q(\theta^{\text{trial}}|\theta^{i-1})\Pr(\theta^{i-1}|D)}\right\}$$

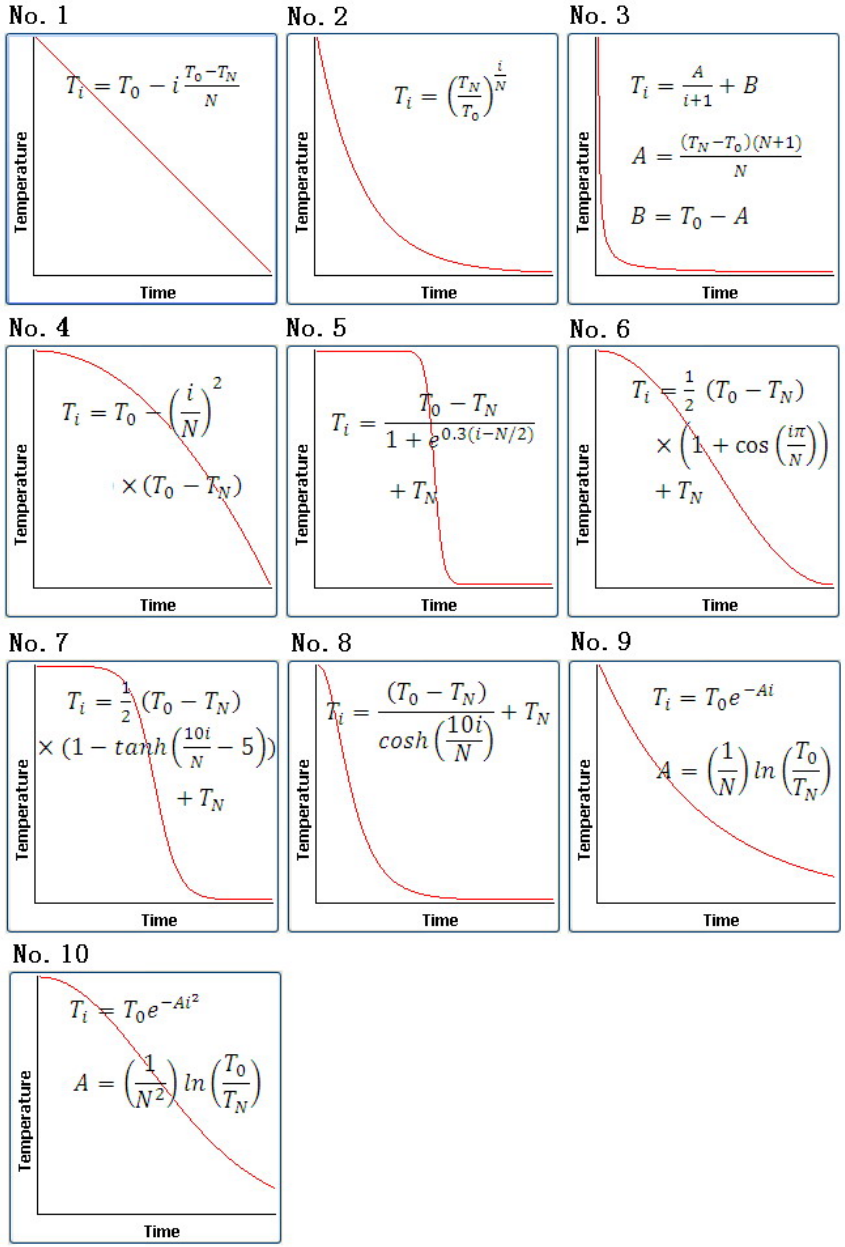
$$= \min\left\{1, \exp\left(-\frac{E(\theta^{\text{trial}}) - E(\theta^{i-1})}{t_i}\right)\right\}$$

MCMC Optimization: Simulated annealing

Algorithm 3 Simulated annealing

procedure $\mathrm{SA}(E, \boldsymbol{\theta})$ ▷ Simulated Annealing
  $\theta^0 \in \pi(\theta)$ ▷ Initialize with values drawn from prior
  for $i = 1, \ldots$ **do**
    $\theta^{\text{trial}} \sim q(\theta^{\text{trial}} | \theta^{i-1})$ ▷ Trial state from proposal distribution q
    $a(\theta^{\text{trial}} | \theta^{i-1}) = \min\left\{1, \exp\left(-\frac{E(\theta^{\text{trial}}) - E(\theta^{i-1})}{T_i}\right)\right\}$ ▷ Acceptance probability
    if $U(0,1) < a(\theta^{\text{trial}} | \theta^{i-1})$ **then**
      $\theta^i \leftarrow \theta^{\text{trial}}$ ▷ Accept the move
    else
      $\theta^i \leftarrow \theta^{i-1}$ ▷ State is unchanged
    end if
  end for
end procedure

MCMC optimization: possible temperature schedules



MCMC inference: parallel tempering

- Setting a temperature schedule is difficult and tedious; we can replace it with multiple chains working at fixed temperatures, following a temperature ladder $0 = t_1 < \ldots < t_N = 1$
- High temperature chains handle the posterior exploration
- Lower temperature chains try to converge on zones of high probability
- Chains are mixed regularly but infrequently. We exchange their states with a Metropolis-Hasting move:

$$a(m \to n) = \min\left\{1, \frac{p_n(\theta_m)}{p_n(\theta_n)} \frac{p_m(\theta_n)}{p_m(\theta_m)}\right\}$$

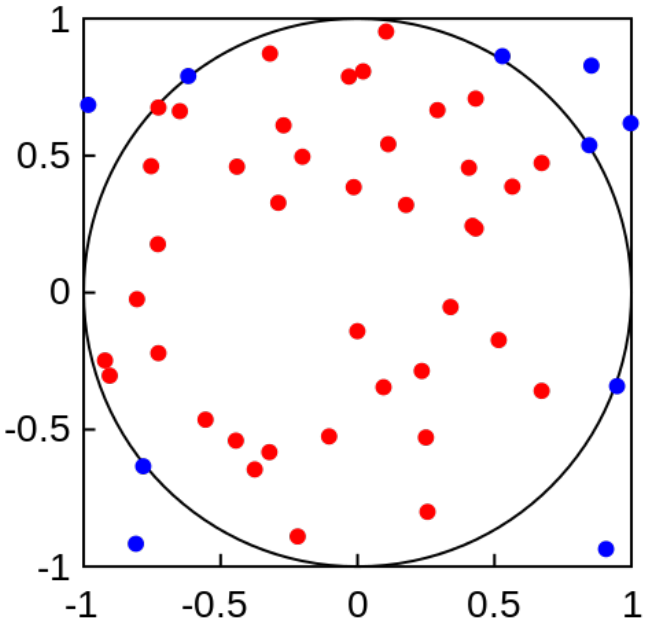
- $p_n = \Pr(\theta|D)^{\frac{1}{t_n}} \to \log a = (\frac{1}{t_n} - \frac{1}{t_m})\left[\log \Pr(\theta_m|D) - \log \Pr(\theta_n|D)\right]$
- Parallel tempering is well-suited to CPU/GPU parallel architectures.

MCMC inference: parallel tempering

Algorithm 4 Parallel tempering

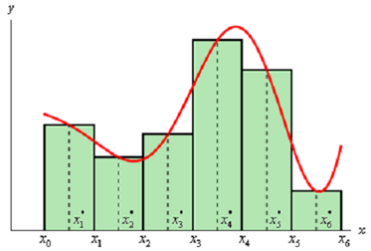
procedure $\mathrm{PT}(E, \theta)$
  $\theta^0 \in \pi(\theta)$ ▷ initialize with values drawn from prior
  $0 = t_1 < \ldots < t_N = 1$ ▷ choose a fixed temperature ladder
  for $i = 1, \ldots$ **do** ▷ global MCMC iterations
    for $n = 1 \ldots N$ **do** ▷ can be done in parallel
      run SA chain n with temp. t_n ▷ can do several iterations
    end for
    $a(E_m|E_n) = \min\left\{1, \exp\left(\left[\frac{1}{t_n} - \frac{1}{t_m}\right](E_n - E_m)\right)\right\}$
    if $U(0,1) < a(E_m|E_n)$ **then**
      exchange state chains ▷ e.g. exchange t_n and t_m
    end if
  end for
end procedure

Monte Carlo Numerical integration: computing π



- Bored on the beach? Use sand to compute π!
- On a sheet of paper, draw a square and its inscribed circle
- Throw sand on your drawing, sand lands randomly on the surface
- The ratio of the number of sand grains in the circle over the number in the square is equal to their surface ratio, $\frac{\pi R^2}{4R^2} = \frac{\pi}{4}$

Numerical integration: classic integration



- E.g. midpoint rule splits the integration domain into N pieces:

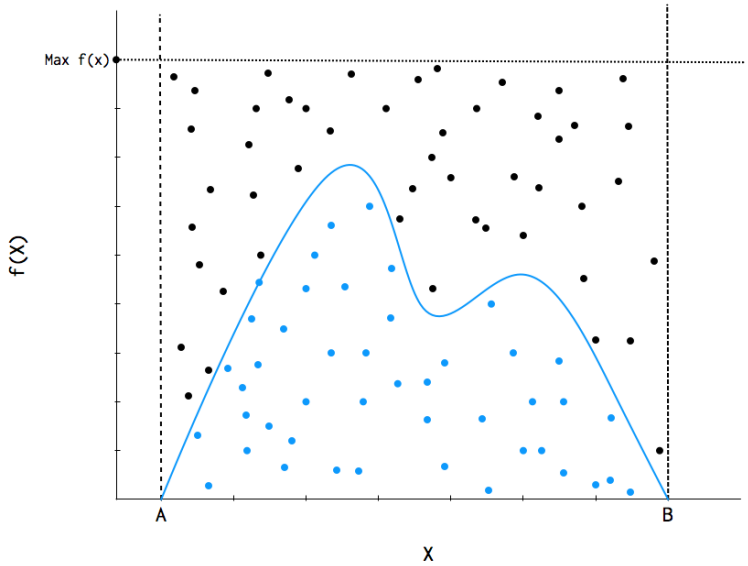
$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{n=0}^{N-1} f\left(\frac{x_n + x_{n+1}}{2}\right) \text{ in 1D}$$

$$\int_{a_1}^{b_1}\dots\int_{a_d}^{b_d} f(x_1\dots x_d)dx_1\dots dx_d = \left(\prod_{k=1}^{d}\frac{b_k-a_k}{N_k}\right)\times$$

$$\sum_k\sum_n f\left(\frac{x_{1n}+x_{1n+1}}{2},\dots,\frac{x_{dn}+x_{dn+1}}{2}\right) \text{ in nD}$$

- Alternatives: trapezoidal rule, Simpson's rule
- Can be generalized in d dimensions
- Error on a d-dimensional integral goes as $O(N^{-\frac{2}{d}})$

Numerical integration: MCMC integration



- N coordinate vector points $\boldsymbol{x_1}, \ldots \boldsymbol{x_N}$ randomly distributed in a d-dimensional volume $[a_1, b_1] \times \ldots \times [a_d, b_d]$
- Let's define $\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i)$ and $\bar{f^2} = \frac{1}{N} \sum_{i=1}^{N} (f(\boldsymbol{x}_i))^2$
- Then $\int_{a_1}^{b_1} \ldots \int_{a_d}^{b_d} f(\boldsymbol{x}) d\boldsymbol{x} = \prod_{i=1}^{d} (b_i - a_i) \bar{f}$: MCMC integration
- Error $\simeq \prod_{i=1}^{d} (b_i - a_i) \sqrt{\frac{\bar{f^2} - (\bar{f})^2}{N}}$, goes as $O(N^{-\frac{1}{2}})$.

Numerical integration: examples

- Given $f(x, y, z) = 4 - x^2 - y^2 - z^2$,

$$\int_0^{0.9} \left(\int_0^1 \left(\int_0^{1.1} f(x, y, z) dz \right) dy \right) dx = 2.9634$$

- Try to get this result using MCMC integration and midpoint rule, compare their errors
- Same for $f(x, y, z, u, v) = \sqrt{6 - x^2 - y^2 - z^2 - u^2 - v^2}$,

$$\int_0^{0.7} \left(\int_0^{0.8} \left(\int_0^{0.9} \left(\int_0^1 \left(\int_0^{1.1} f(x, y, z, u, v) dv \right) du \right) dz \right) dy \right) dx \simeq 1.18878359$$

- MCMC integration error is lower than that other methods in higher dimensions: in fact it is the only generic way of integrating reliably.