

Title: Fitting models to data using Markov Chain Monte Carlo

Speakers: Dustin Lang

Collection: Symmetries Graduate School 2023

Date: January 23, 2023 - 2:00 PM

URL: <https://pirsa.org/23010076>

Markov Chain Monte Carlo

Dustin Lang
Perimeter Institute for Theoretical Physics

Symmetries Graduate School, 2023-01-23

Borrowing heavily from Dan Foreman-Mackey's slides
<https://speakerdeck.com/dfm/data-analysis-with-mcmc1>

These slides are available at
<https://github.com/dstndstn/MCMC-talk>

1

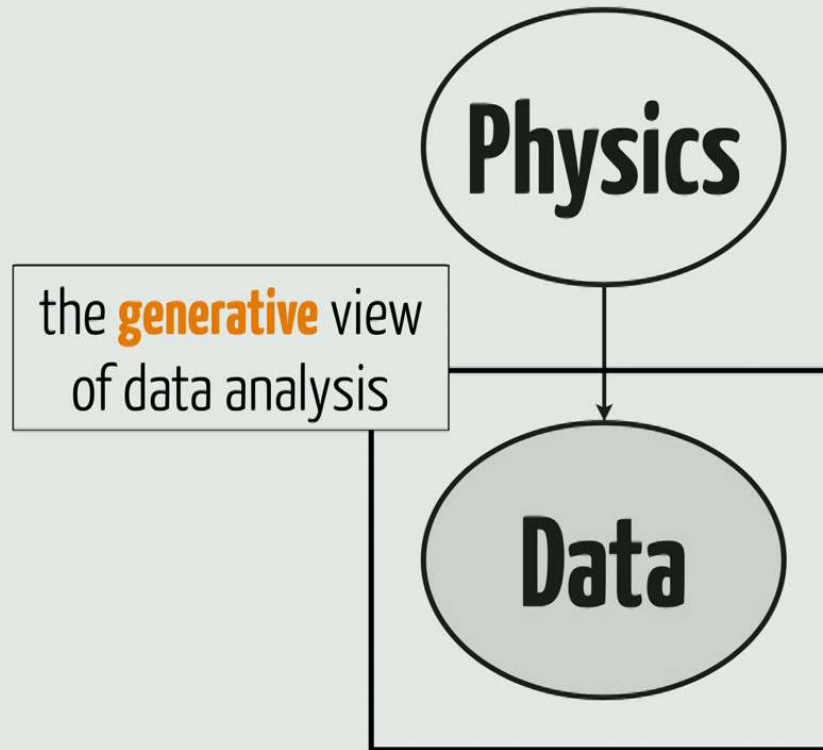
data analysis with

Markov chain Monte Carlo

Dan Foreman-Mackey

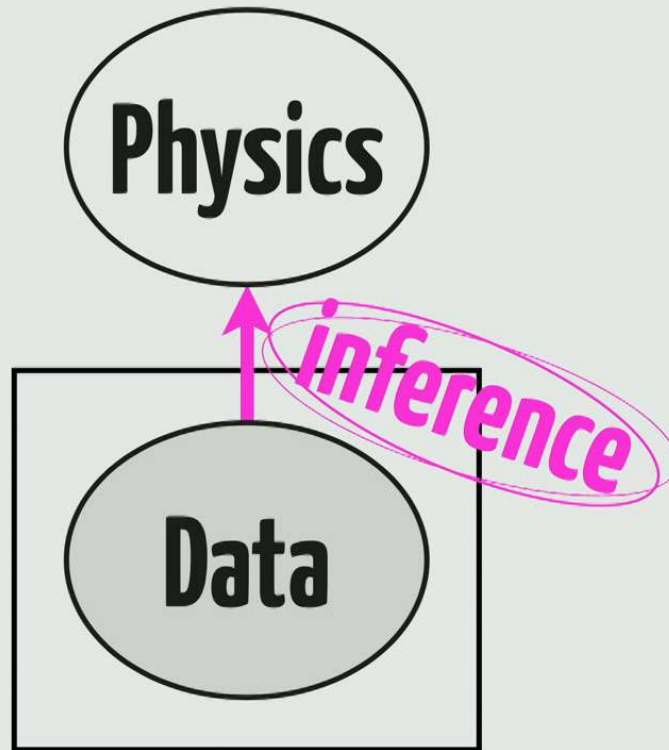
CCPP@NYU

2



a sketch of
The graphical model of my research.

3



a sketch of
The graphical model of my research.

4

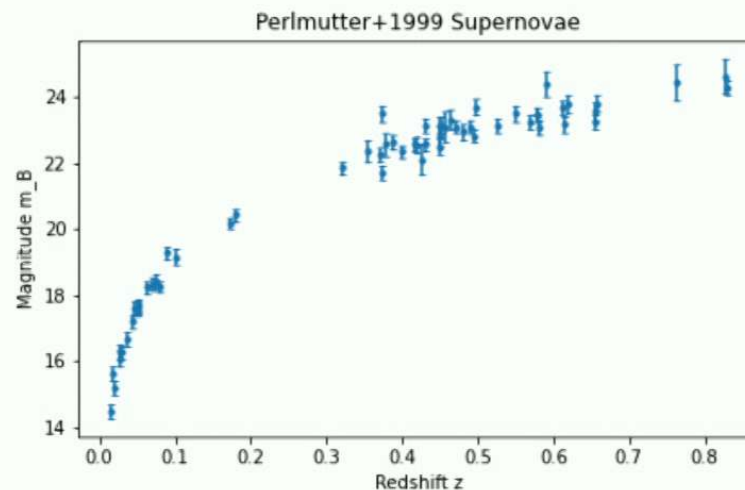
$p(\text{data} \mid \text{physics})$
likelihood function/generative model



$p(\text{physics} \mid \text{data}) \propto p(\text{physics}) p(\text{data} \mid \text{physics})$
posterior probability

An example

- ▶ Perlmutter+1999 (<https://arxiv.org/abs/astro-ph/9812133>)
- ▶ Measured the observed peak brightnesses of a sample of type-1a supernovae (in astronomer “mag” units), and the redshifts (“z”) of the supernova host galaxies
- ▶ $\text{mag} = \text{mag}_{\text{intrinsic}} + \text{luminosity_distance}(z, \text{parameters})$



An example

- ▶ Generative model:

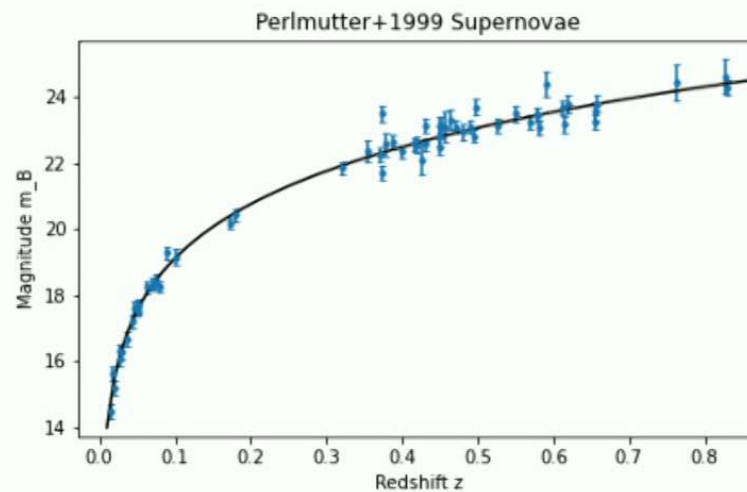
$$\text{mag}_i = \text{mag}_{\text{intrinsic}} + \text{luminosity_distance}(z_i, \text{parameters}) + \epsilon_i$$

- ▶ Probability of data given a model (“likelihood”):

$$p(\{\text{mag}_i\} | \text{params}) = \prod_i \text{Gaussian}(\text{mag}_i | \mu = f(z_i, \text{params}), \sigma_i^2)$$

- ▶ $p(\text{mag}_i | \Omega_M, \Omega_\Lambda) = \mathcal{N}(\text{mag}_i | \text{mag}_{\text{int}} + D_L(z_i, \Omega_M, \Omega_\Lambda), \sigma_i^2)$

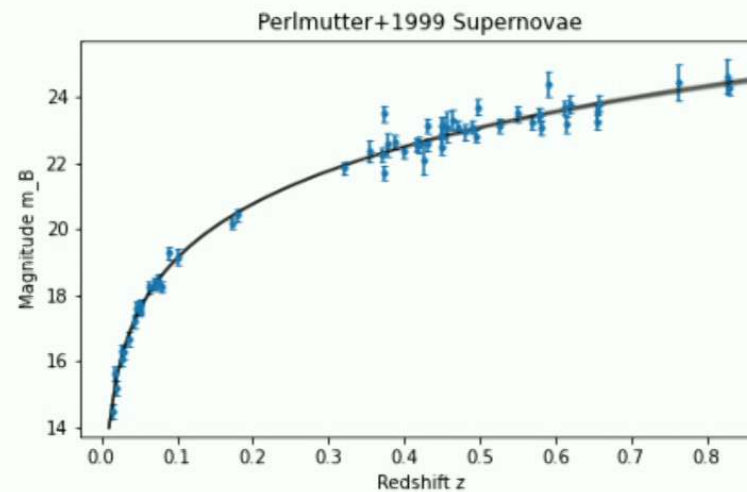
I



7

An example

- ▶ Use Bayes' theorem to convert data likelihoods into constraints on the model parameters $\theta = \{\Omega_M, \Omega_\Lambda\}$
- ▶ $p(\theta | \text{data}) \propto p(\theta) p(\text{data} | \theta)$
- ▶ $p(\Omega_M, \Omega_\Lambda | \{\text{mag}_i\}) \propto p(\Omega_M, \Omega_\Lambda) \prod_i \mathcal{N}(\text{mag}_i | \text{mag}_{\text{int}} + D_L(z_i, \Omega_M, \Omega_\Lambda), \sigma_i^2)$

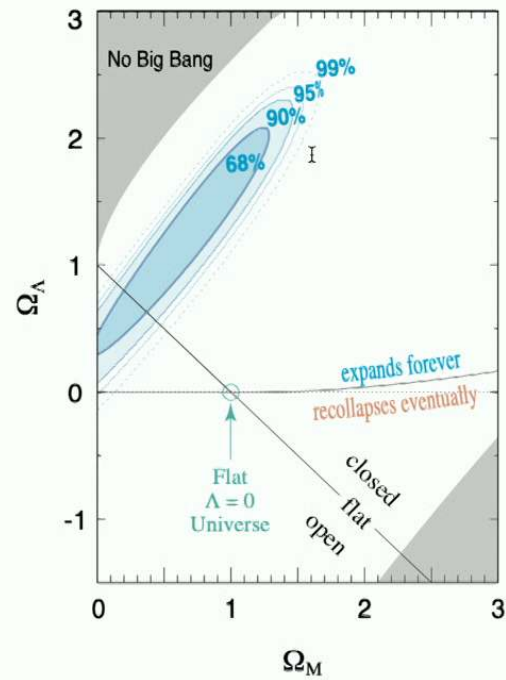


I

8

An example

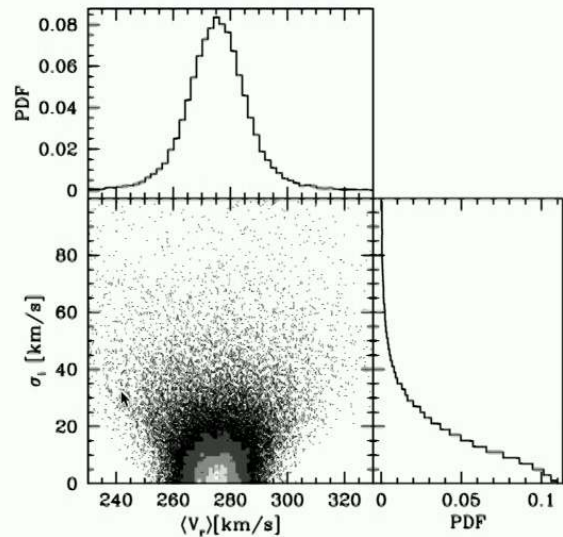
- ▶ Resulting parameter constraints (blue ellipse):



Why we often need MCMC

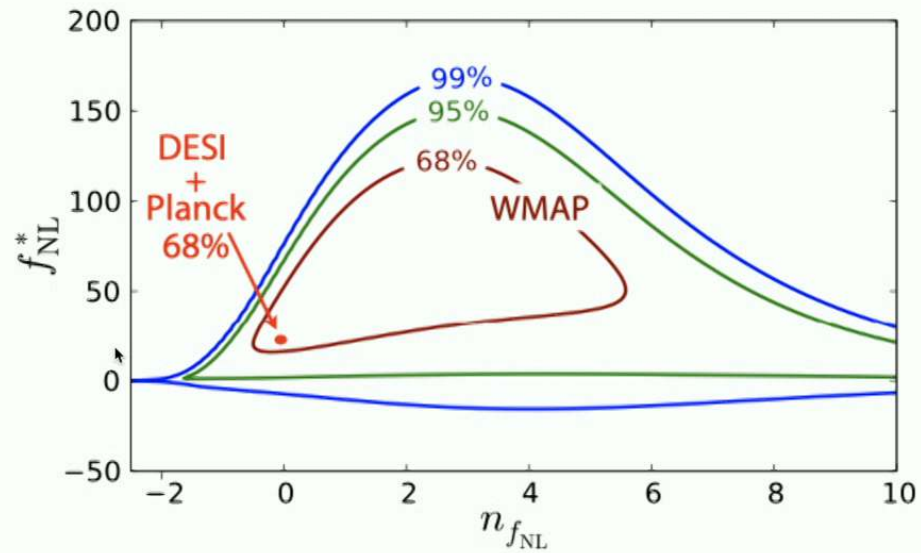
- ▶ Real-life models and likelihoods are often complex
- ▶ ... so the resulting **constraints** have complicated distributions (not Gaussians!)
- ▶ ... but we can represent them with **samplings**
- ▶ MCMC is used for drawing samples from probability distributions that we can compute numerically but cannot solve analytically

Samplings to represent constraints - examples



- ▶ From <https://arxiv.org/abs/1910.04899>
- ▶ With a sampling: **Marginalize** over a parameter by projecting it out

Samplings to represent constraints - examples



- ▶ From <https://arxiv.org/abs/1611.00036>

MCMC

draws samples from a probability function

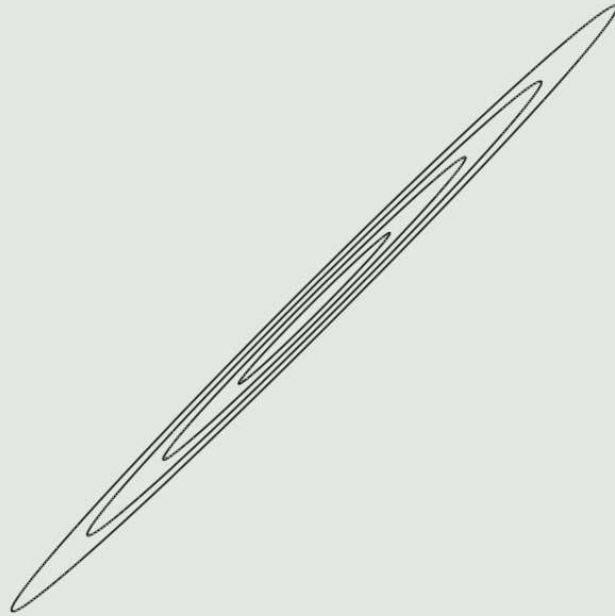
and all you need to be able to do is

evaluate

the function

(up to a constant)

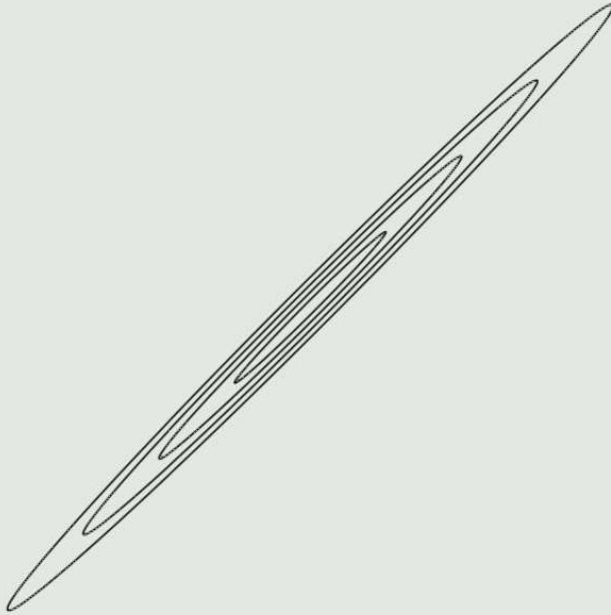
13



Metropolis-Hastings

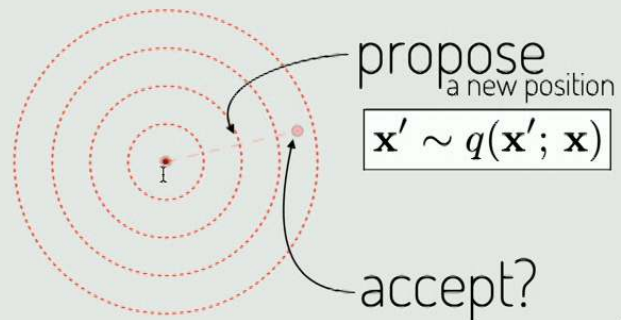
14

start here
perhaps



Metropolis-Hastings
in an ideal world

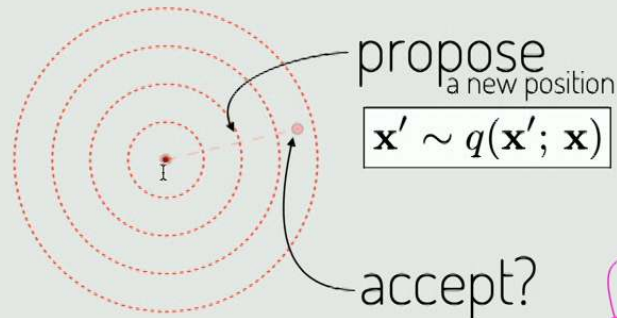
16



$$p(\text{accept}) = \min \left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')} \frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})} \right)$$

Metropolis-Hastings
in an ideal world

18



accept?

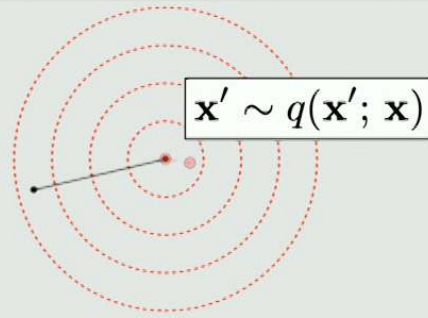
only **relative** probabilities

$$p(\text{accept}) = \min \left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')} \frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})} \right)$$

definitely.

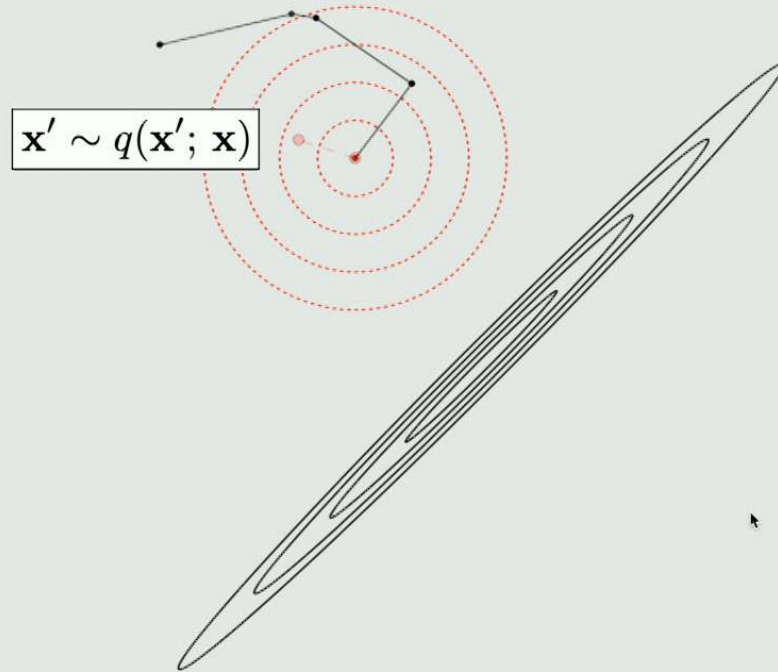
Metropolis-Hastings
in an ideal world

20



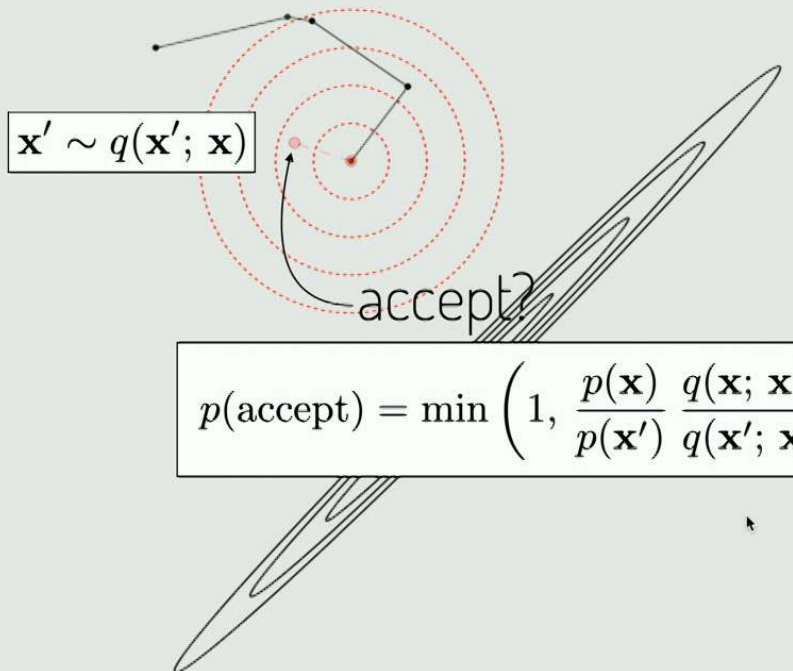
Metropolis-Hastings
in an ideal world

21



Metropolis-Hastings

in an ideal world

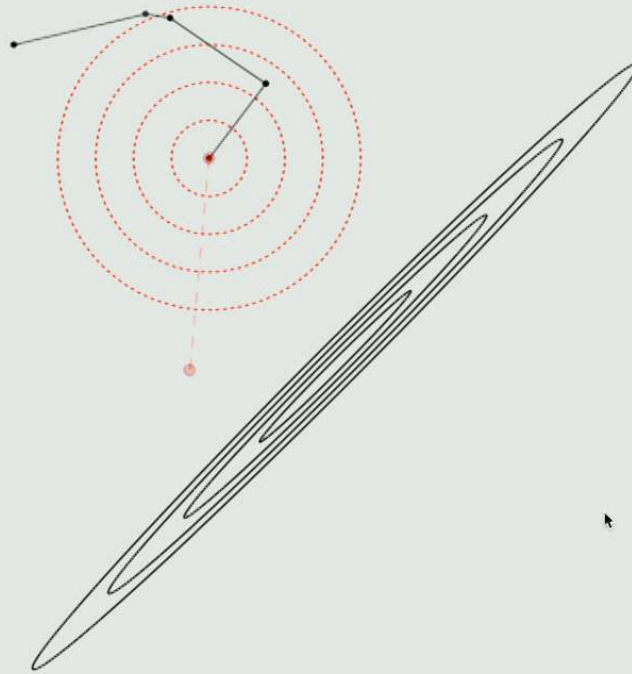

$$\mathbf{x}' \sim q(\mathbf{x}'; \mathbf{x})$$

accept?

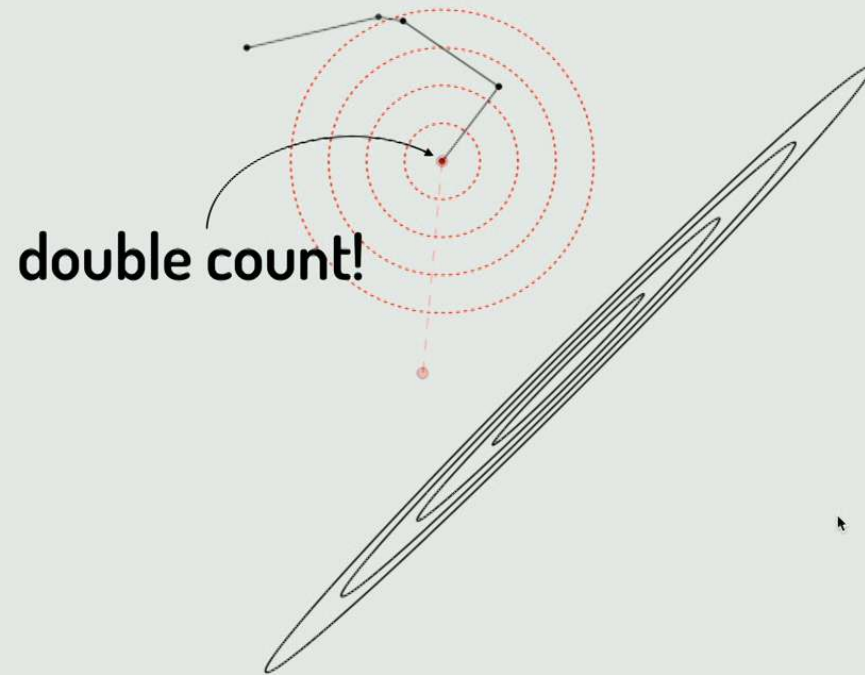
$$p(\text{accept}) = \min \left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')} \frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})} \right)$$

Metropolis–Hastings
in an ideal world

23

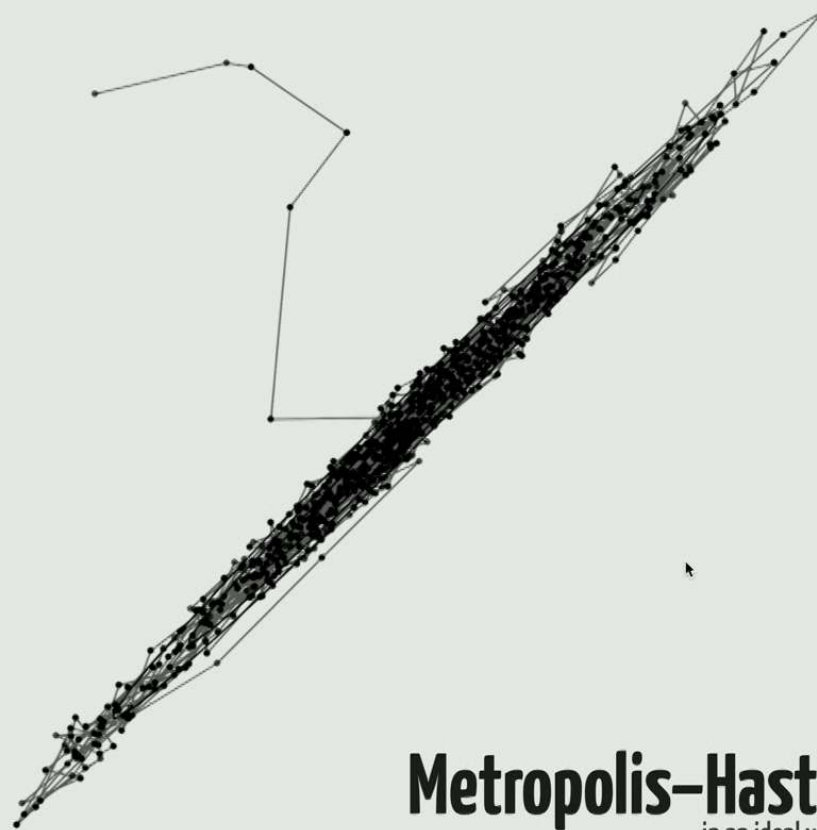


Metropolis-Hastings
in an ideal world



double count!

Metropolis-Hastings
in an ideal world



Metropolis-Hastings

in an ideal world

28

About the name

- ▶ **Monte Carlo**: a reference to the famous Monte Carlo Casino in Monaco, alluding to the randomness used in the algorithm
- ▶ **Markov Chain**: a list of samples, where each one is generated by a process that only looks at the previous one.
- ▶ **Markov**: a 19th-century Russian mathematician and impressive-moustache-haver with an [extensive list of things named after him](#)
- ▶ **Metropolis–Hastings**: lead authors of 1953 and 1970 papers (resp.) giving the algorithm with symmetric and general proposal distributions (resp.)

The Algorithm (1)

```
def mcmc(prob_func, propose_func, initial_pos, nsteps):
    p = initial_pos
    prob = prob_func(p)
    chain = []
    for i in range(nsteps):
        # propose a new position in parameter space
        # ...
        # compute probability at new position
        # ...
        # decide whether to jump to the new position
        if # ...
            # ...
            # ...
        # save the position
        chain.append(p)
    return chain
```

30

The Algorithm (2)

```
def mcmc(prob_func, propose_func, initial_pos, nsteps):
    p = initial_pos
    prob = prob_func(p)
    chain = []
    for i in range(nsteps):
        # propose a new position in parameter space
        p_new = propose_func(p)
        # compute probability at new position
        prob_new = prob_func(p_new)
        # decide whether to jump to the new position
        if prob_new / prob > uniform_random():
            p = p_new
            prob = prob_new
        # save the position
        chain.append(p)
    return chain
```

31

The Algorithm (3)

```
def mcmc(logprob_func, propose_func, initial_pos, nsteps):
    p = initial_pos
    logprob = logprob_func(p)
    chain = []
    for i in range(nsteps):
        # propose a new position in parameter space
        p_new = propose_func(p)
        # compute probability at new position
        logprob_new = logprob_func(p_new)
        # decide whether to jump to the new position
        if exp(logprob_new - logprob) > uniform_random():
            p = p_new
            logprob = logprob_new
        # save the position
        chain.append(p)
    return chain
```

32

The Algorithm (4)

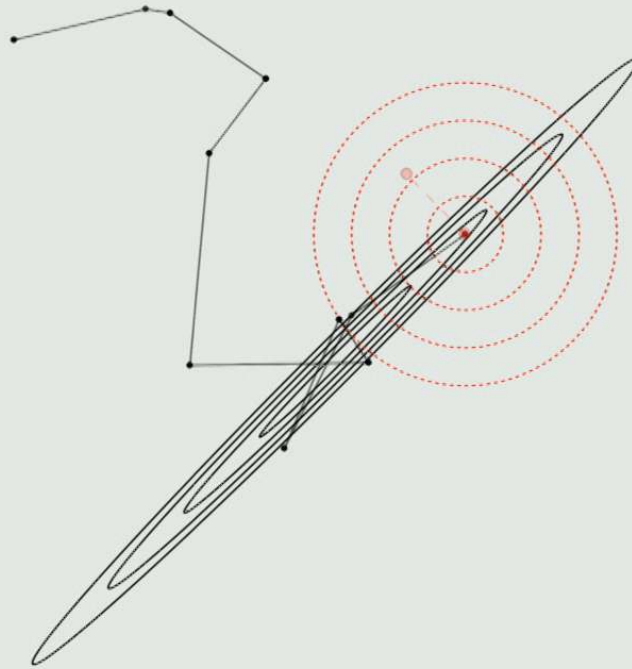
```
def mcmc(logprob_func, propose_func, initial_pos, nsteps):
    p = initial_pos
    logprob = logprob_func(p)
    chain = []
    naccept = 0
    for i in range(nsteps):
        # propose a new position in parameter space
        p_new = propose_func(p)
        # compute probability at new position
        logprob_new = logprob_func(p_new)
        # decide whether to jump to the new position
        if exp(logprob_new - logprob) > uniform_random():
            p = p_new
            logprob = logprob_new
            naccept += 1
        # save the position
        chain.append(p)
    return chain, naccept/nsteps
```

33

Practicalities

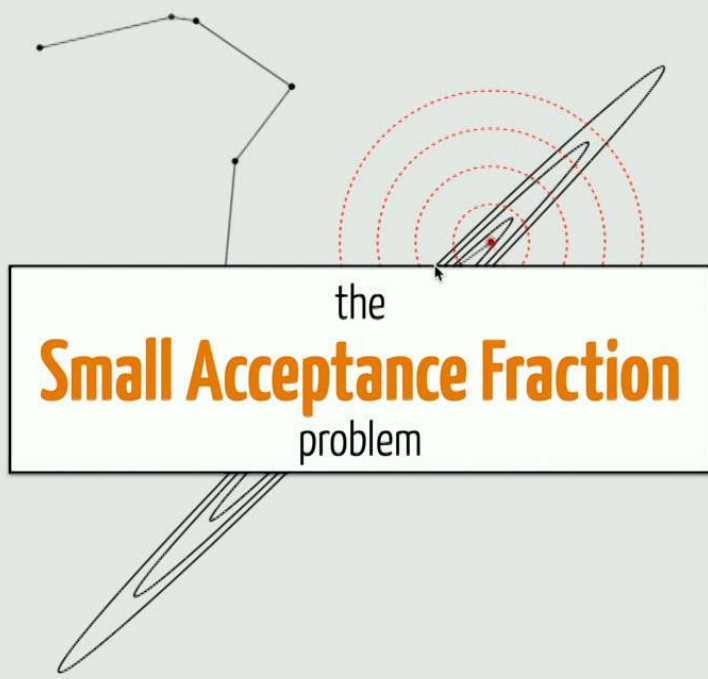
- ▶ How do I choose a proposal distribution?
- ▶ How many steps do I have to take?

I



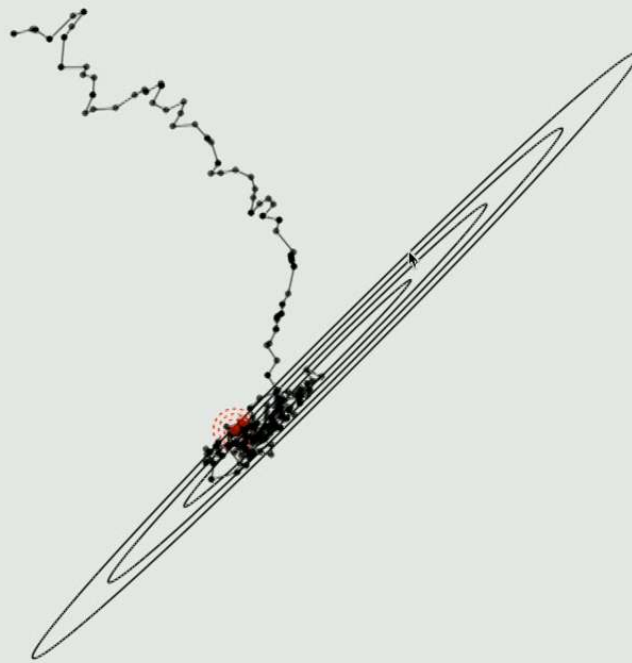
Metropolis-Hastings
in the real world

35



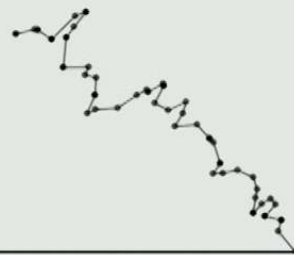
the
Small Acceptance Fraction
problem

Metropolis–Hastings
in the real world



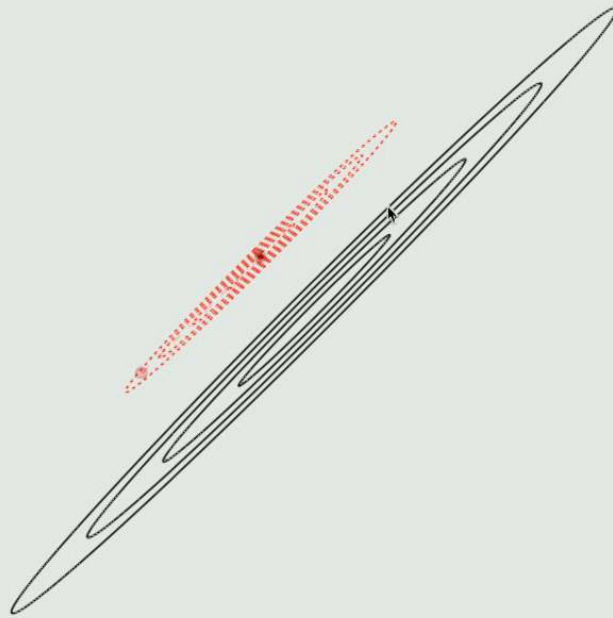
Metropolis-Hastings
in the real world

42



the
Huge Acceptance Fraction
problem

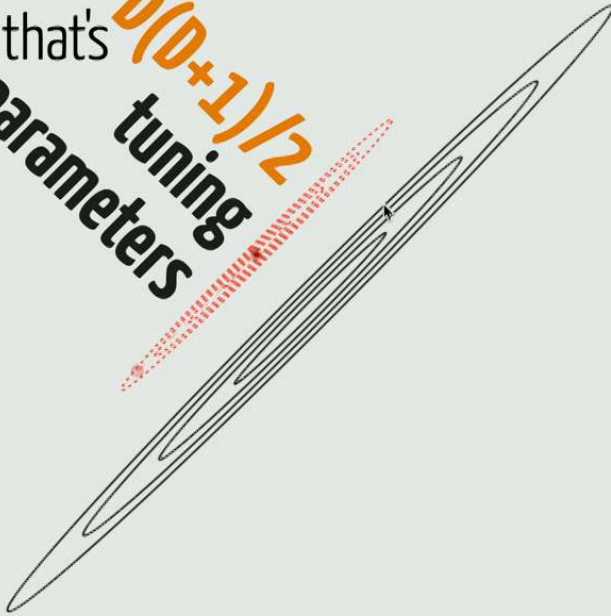
Metropolis–Hastings
in the real world



Metropolis-Hastings
in the real world

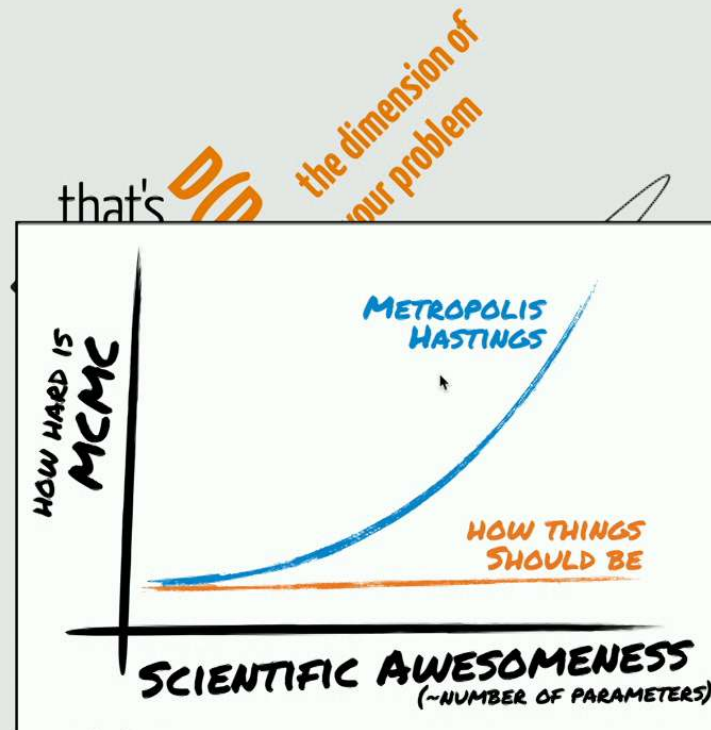
44

that's $O(D+1)/2$
tuning
parameters



Metropolis-Hastings
in the real world

45



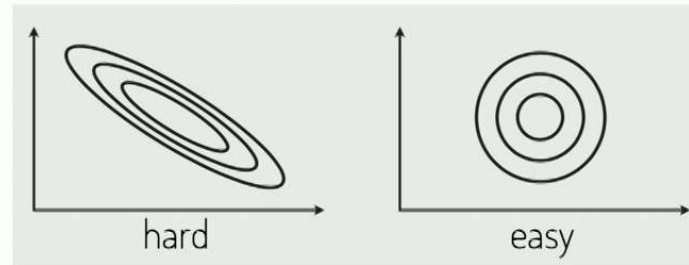
Metropolis-Hastings

in the real world

47

A connection to symmetries

- ▶ In Metropolis–Hastings MCMC, the *proposal distribution* needs **tuning parameters**, especially as dimensionality increases
- ▶ Can be seen as a lack of **symmetry** in the algorithm—the algorithm is sensitive to the parameterization of the problem
- ▶ For example, it's not invariant to an **affine** transformation
- ▶ **Next lecture**, I'll show you an alternative algorithm that **does** have affine invariance



How many samples do I need?

- ▶ Burn-in — skip the first N samples
- ▶ *Has my chain converged?*
- ▶ MCMC produces **correlated** samples, so
 - ▶ How correlated are my samples?
 - ▶ Can measure the *autocorrelation time* τ
 - ▶ Keep $1/\tau$ of the MCMC samples
 - ▶ eg <https://github.com/dfm/acor>
 - ▶ How many uncorrelated samples do I need?
 - ▶ No easy general answer to this question!
 - ▶ “How many can you afford?”

How many samples do I need?

- ▶ Burn-in — skip the first N samples
- ▶ *Has my chain converged?*
- ▶ MCMC produces **correlated** samples, so
 - ▶ How correlated are my samples?
 - ▶ Can measure the *autocorrelation time* τ
 - ▶ Keep $1/\tau$ of the MCMC samples
 - ▶ eg <https://github.com/dfm/acor>
 - ▶ How many uncorrelated samples do I need?
 - ▶ No easy general answer to this question!
 - ▶ “How many can you afford?”

as you learned in middle school

$$\int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N} \sum_{n=1}^N f(\mathbf{x}_n)$$

where: $\mathbf{x}_n \sim p(\mathbf{x})$

error: $\delta \propto \frac{1}{\sqrt{N}}$

number of
independent
samples

52

Conclusions

- ▶ MCMC remains an essential tool for probabilistic inference
- ▶ For science: lets us constrain model parameters based on data (Bayesian inference)
- ▶ Beguilingly simple algorithm, but difficult practicalities
- ▶ MCMC has beautiful theoretical guarantees... as compute time $\rightarrow \infty$