Bayesian Approaches to Inverse Problems in Astrophysics and Cosmology

lecture 3

Dr. Prashin Jethwa
Institut für Astrophysik

Lecture 3

- Deep-dive into the “gold-standard” inference algorithm: HMC/NUTS
- Gaussian Processes (GPs)

By the end of this lecture you should be able to:

- Describe what the HMC/NUTS algorithm does
    - specifically with reference to: step-size, target_acceptance_probability, warm-up stage, divergent transitions
- Explain the convergence diagnostics: N_eff, R_hat, N_divergences
- Suggest strategies to fix HMC/NUTS inference with poor diagnostics

- Understand what it means that GPs are priors over function space
- Use GPs for inference with the JAX-based `tinygp` package

Markov-Chain Monte Carlo Sampling

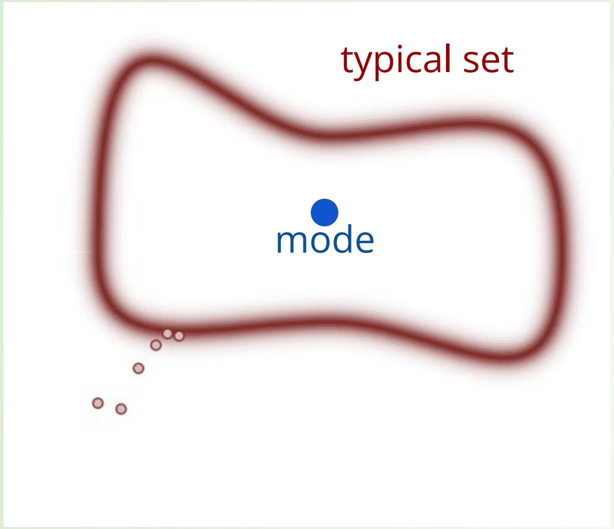
- Reminder from last lecture:

- There are several different types of MCMC algorithm
    - with a different transition kernel $T(\boldsymbol{\theta}_m, \boldsymbol{\theta}_{m+1})$
    - online demonstration

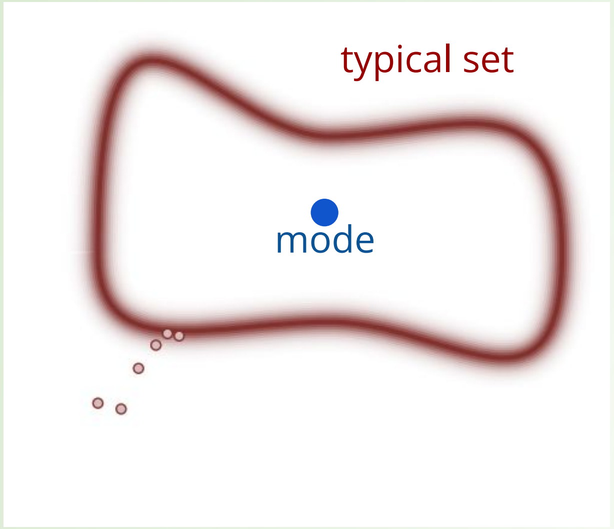
- Compare two:
    - Random Walk Metropolis Hastings
    - HMC/NUTS: Hamiltonian Monte Carlo + No-U-Turn Sampler

Hamiltonian Monte Carlo (HMC)
&
No U-Turn Sampler (NUTS)

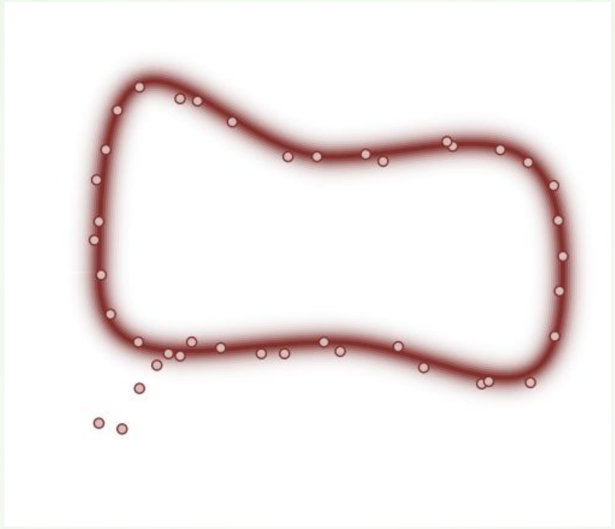
The gold-standard inference algorithm

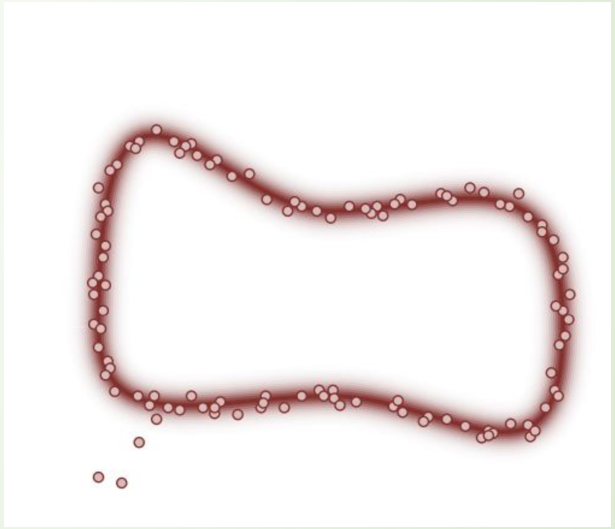


Target distribution: $p(\mathbf{x})$

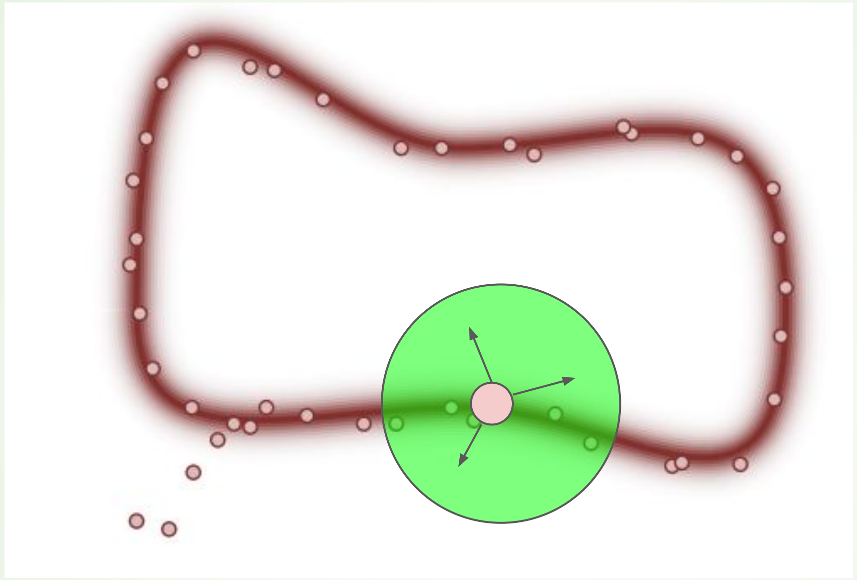


Target distribution: p(**x**)





Goal: construct a MCMC transition operator that will draw many uncorrelated samples from the typical set

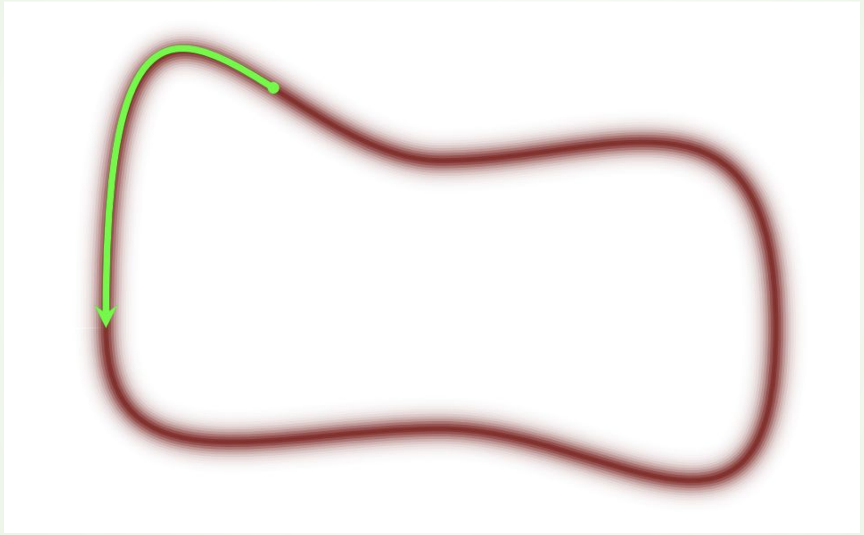


Random Walk Metropolis Hastings

proposes a transition to a Gaussian blob centered around current position.

This becomes very inefficient for high dimensions

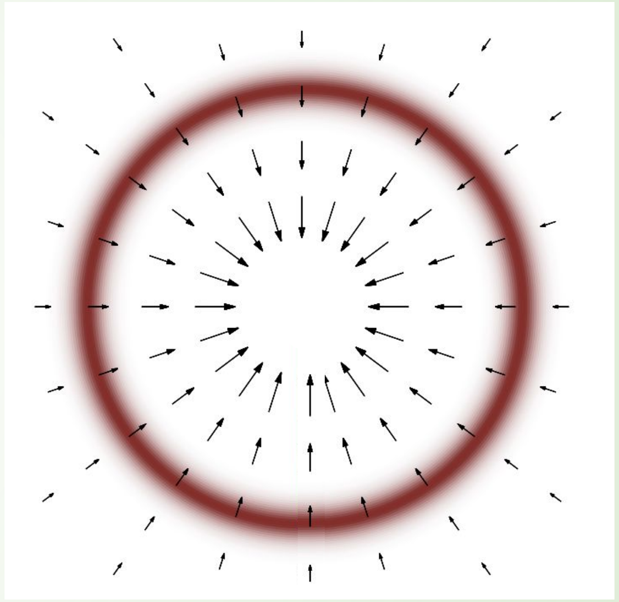
Random Walk Metropolis-Hastings does not scale well to high dimensions.



Solution: use a transition which moves *through* the typical set towards new, unexplored neighborhoods.

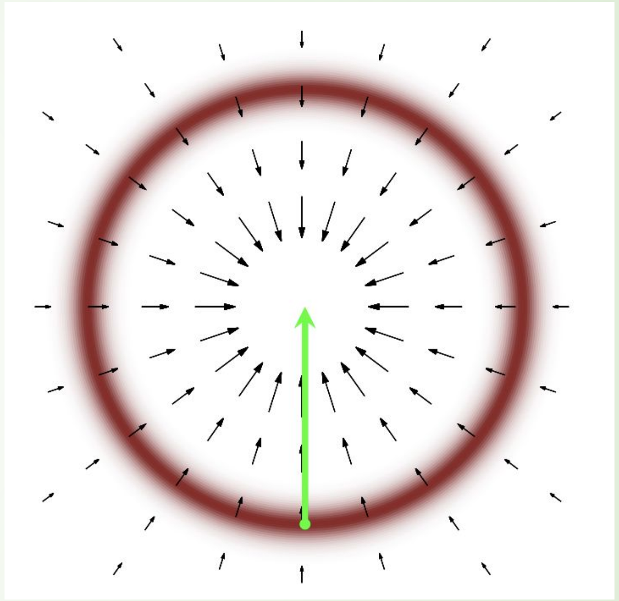
How to transition along the typical set

- Say we have target distribution p(**x**)
- The gradient of the target distribution ∇p(**x**) defines a vector field
- What would happen to a particle which follows a trajectory along the gradient?



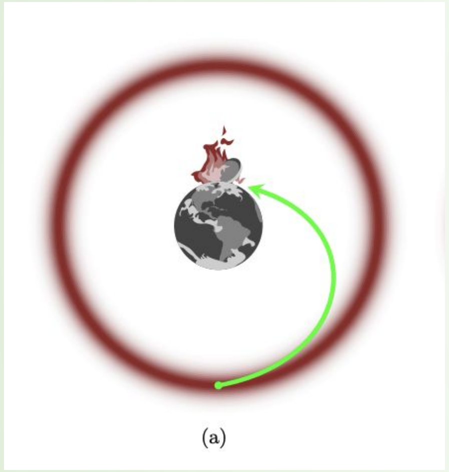
How to transition along the typical set

- Say we have target distribution p(**x**)
- The gradient of the target distribution ∇p(**x**) defines a vector field
- What would happen to a particle which follows a trajectory along the gradient?
    - It will end up at the mode

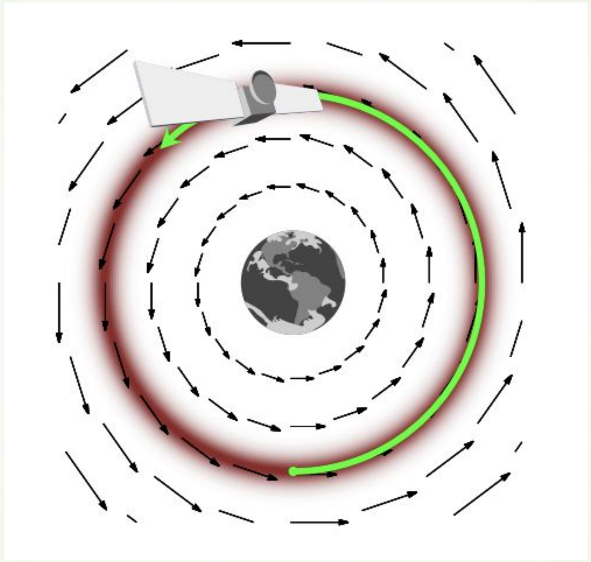


- … to instead move *along* the typical set, we must give the particles some *momentum*

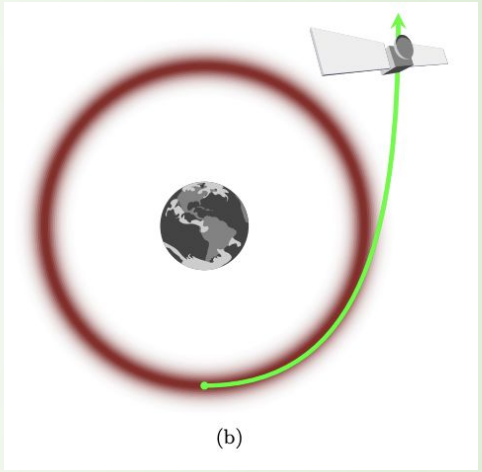
Momentum: physical analogy with a gravitational potential



Too little momentum
→ particle crashes to the mode



Correct momentum
→ particle orbits along the typical set



Too much momentum
→ particle escapes

The *correct* momentum via Hamiltonian Dynamics

- Introduce extra momentum variables $x \to (x, v)$
- Define the joint distribution over position and momentum: $p(x, v) = p(v \mid x)\, p(x)$
- Write this in terms of a Hamiltonian function H $= \exp[-H(x, v)]$

$$\begin{array}{lcccc} \to \quad H(x, v) & = & -\log p(v \mid x) & & -\log p(x) \\ & := & K(v, x) & + & V(x) \\ \text{energy} & := & \text{kinetic energy} & + & \text{potential energy} \end{array}$$

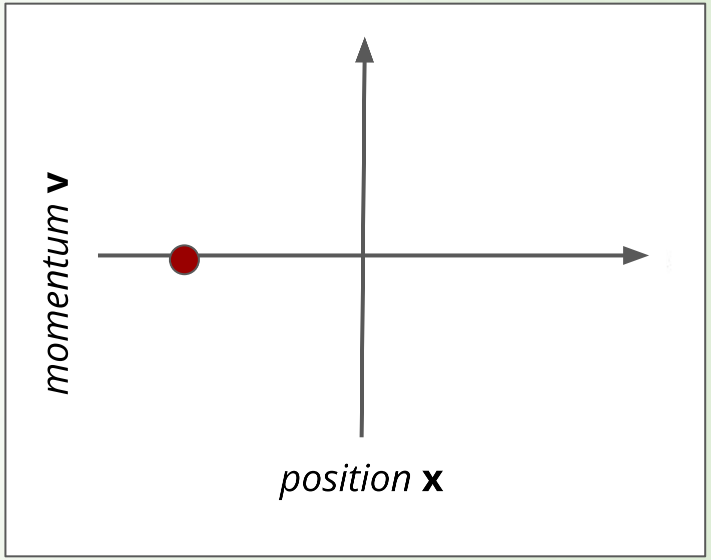
- If we take any initial point (x_0, v_0) and evolve it via Hamilton's equations:

$$dx/dt = +\,\partial H / \partial v$$
$$dv/dt = -\,\partial H / \partial x$$

then the resulting trajectory naturally orbits along the typical set of the target distribution

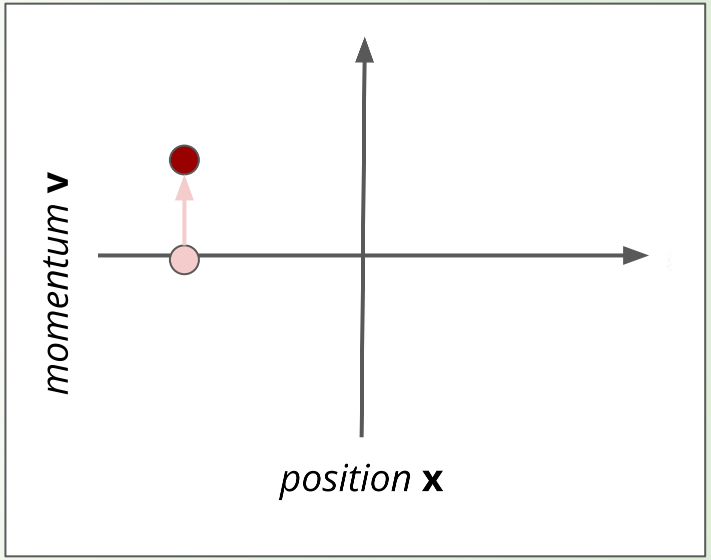
HMC: the algorithm

1. Start at initial point x_n



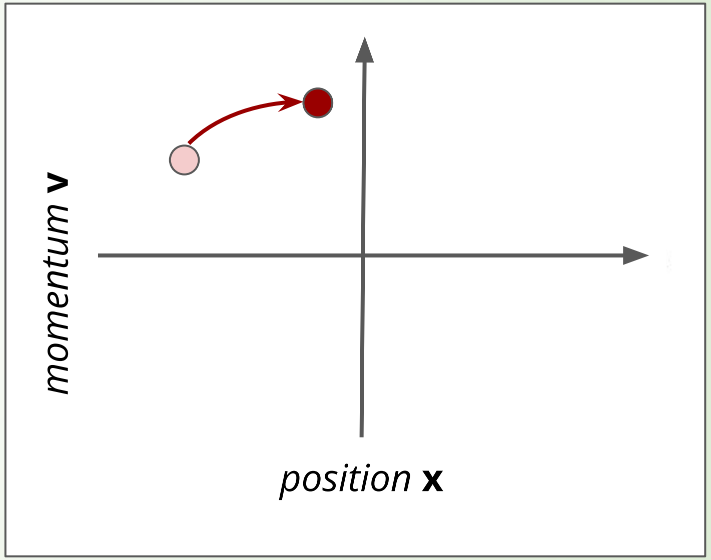
HMC: the algorithm

1. Start at initial point x_n
2. Sample a momentum $v_n \sim p(v \mid x_n)$



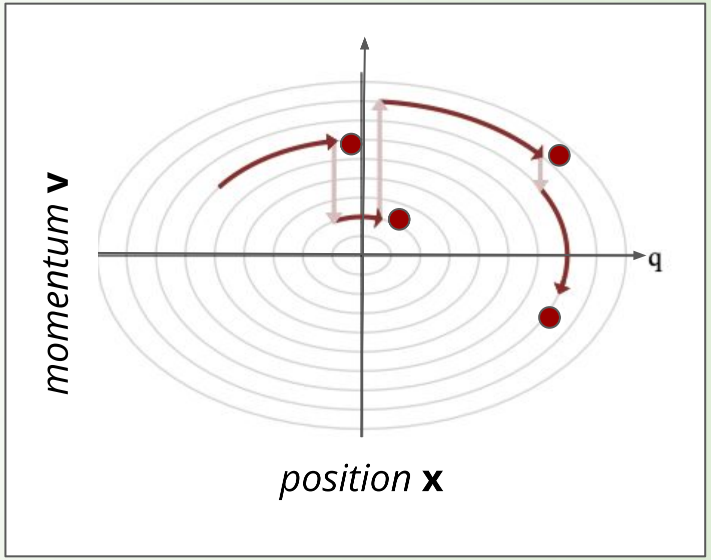
HMC: the algorithm

1. Start at initial point x_n
2. Sample a momentum $v_n \sim p(v \mid x_n)$
3. Use Hamilton's equations to numerically an orbit $(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})$



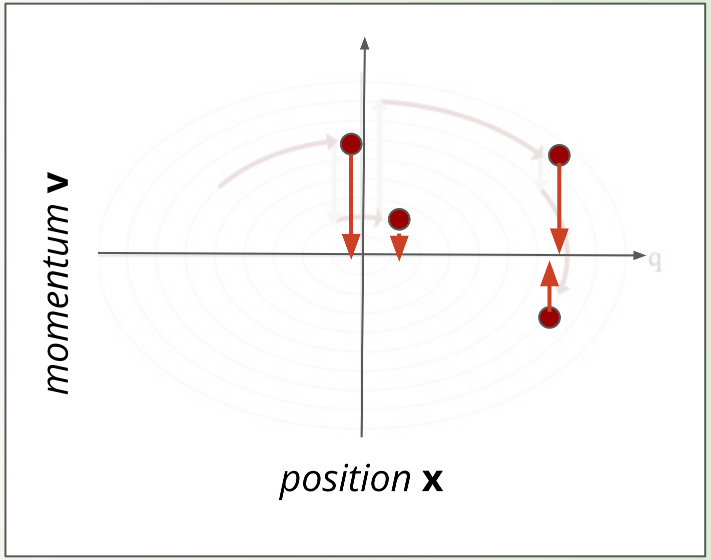
HMC: the algorithm

1. Start at initial point x_n
2. Sample a momentum $v_n \sim p(v \mid x_n)$
3. Use Hamilton's equations to numerically an orbit $(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})$
4. Accept/reject the new point…
5. Repeat for desired number of samples



HMC: the algorithm

1. Start at initial point x_n
2. Sample a momentum $v_n \sim p(v \mid x_n)$
3. Use Hamilton's equations to numerically an orbit $(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})$
4. Accept/reject the new point...
5. Repeat for desired number of samples
6. Project samples down to position space i.e. forget the momenta $(x,v) \rightarrow x$



What is $p(v \mid x)$?

- The choice of $p(v \mid x)$ - i.e. the kinetic energy term in the Hamiltonian - is a free choice - different possibilities exist
- Most common choice: Euclidean-Gaussian Kinetic Energy:

$$p(v \mid x) = N(v \mid 0 , M)$$

i.e. the kinetic energy

$$K(v, x) = - \log p(v \mid x) = \tfrac{1}{2} v^T M v + \log |M|$$

where M is the mass-matrix, i.e. tuning-parameters of the algorithm

- M is tuned to re-scale/rotate the parameter space distribution into a more standardised frame

Numerically integrating the orbit?

- Take an initial point (x0, v0) and evolve it via Hamilton's equations:

$$dx/dt = + \partial H / \partial v$$
$$dv/dt = - \partial H / \partial x := a$$

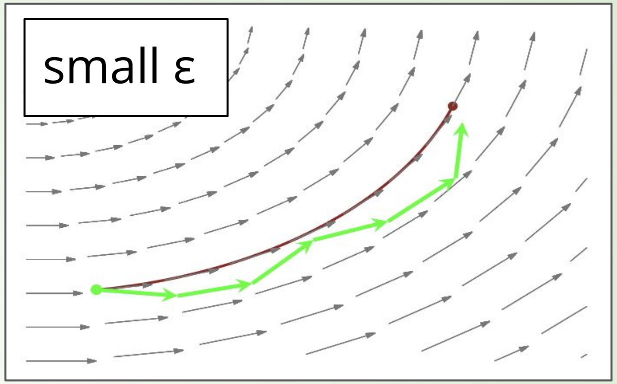
- When discretized, with some step-size **ε** these can be integrated numerically with a leapfrog (AKA kick-drift-kick) integrator:

$$v_{i+½} = v_i + a_i \, \varepsilon/2 \qquad \text{“kick”}$$
$$x_{i+1} = x_i + v_{i+½} \, \varepsilon \qquad \text{“drift”}$$
$$v_{i+1} = v_{i+½} + a_{i+1} \, \varepsilon/2 \qquad \text{“kick”}$$

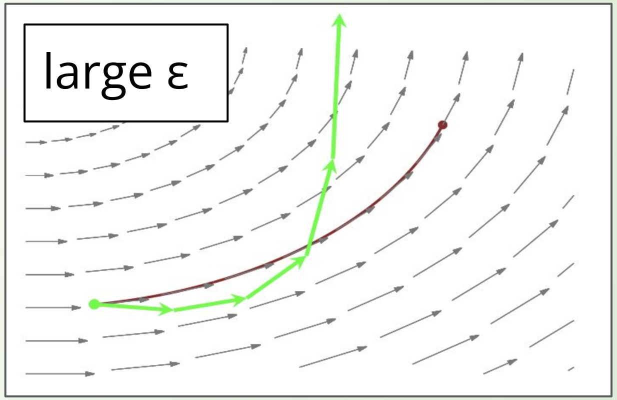
- This integrator is symplectic - i.e. it preserves energy - in contrast to other schemes e.g. Runga-Kutta methods

Setting the step-size ε

- To efficiently make large trajectories, we want a large step-size ε
- But if ε is too large, energy is no longer conserved
  - the trajectory diverges from the true one



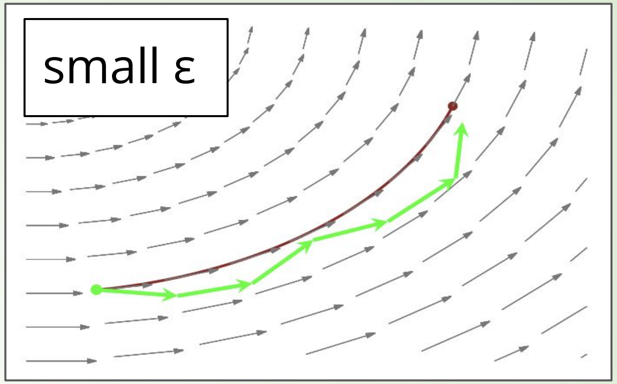
Numerical orbit stays near truth



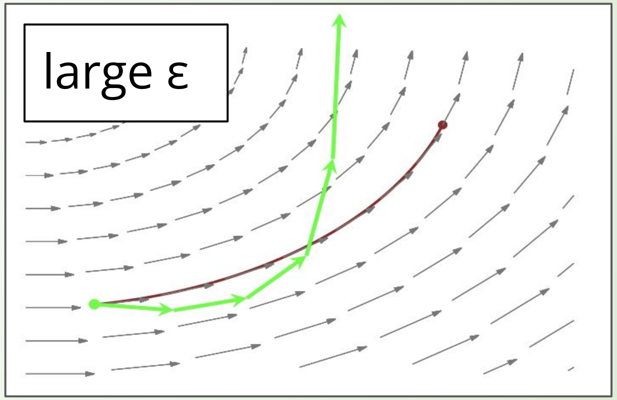
Numerical orbit diverges from truth

Setting the step-size ε

- To efficiently make large trajectories, we want a large step-size ε
- But if ε is too large, energy is no longer conserved
    - the trajectory diverges from the true one
- We can identify these cases by looking at energy drift between the trajectory endpoints:
    - $\Delta E = H(x_0, v_0) - H(x_n, v_n)$
    - If $\Delta E >$ some threshold, the trajectory is divergent
- Gives an extra step in HMC algorithm...



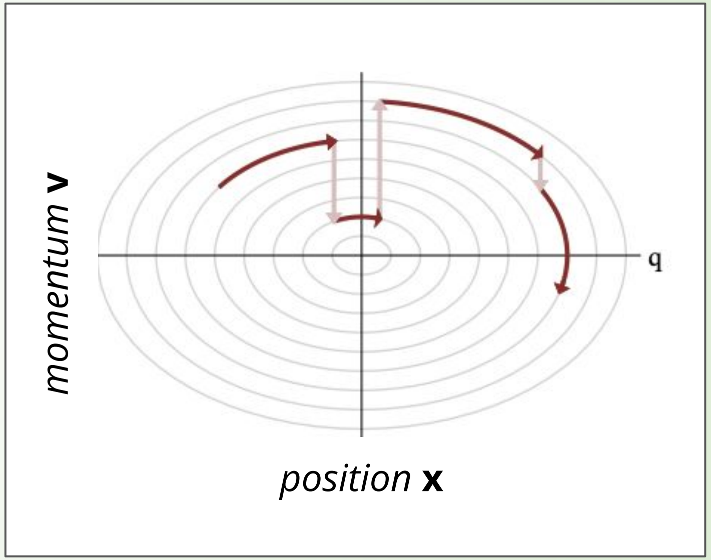
Numerical orbit stays near truth



Numerical orbit diverges from truth

HMC: the algorithm

1. Start at initial point x_n
2. Sample a momentum $v_n \sim p(v \mid x_n)$
3. Use Hamilton's equations to numerically an orbit $(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})$
4. Accept/reject new point based on energy drift in trajectory:
   a. Large drift → less likely to be accepted
   b. Implemented in a way which satisfies the detailed-balance equation
5. Repeat for desired number of samples
6. Project samples down to position space i.e. forget the momenta $(x,v) \rightarrow x$

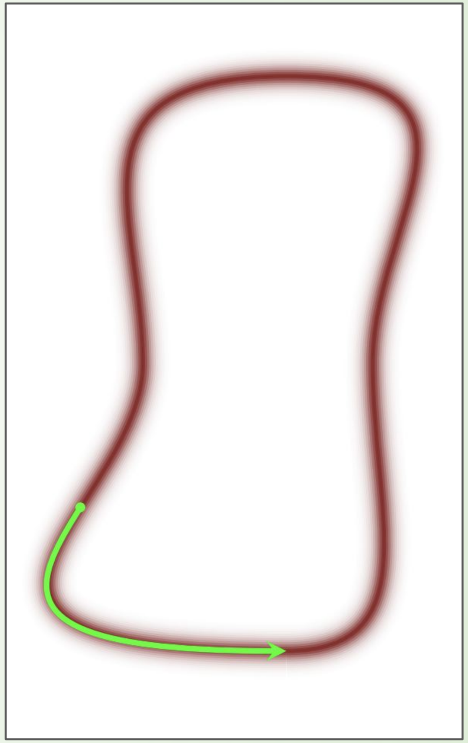


No U-Turn Sampler (NUTS)

- As well as a step-size, we need to know *how many* steps to take

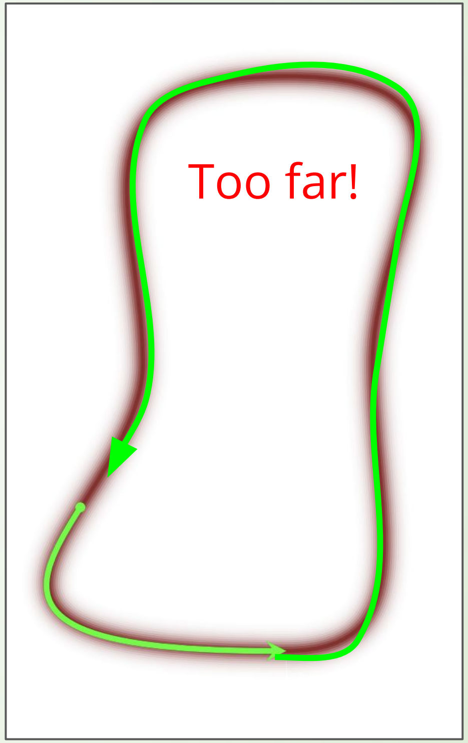
No U-Turn Sampler (NUTS)

- As well as a step-size, we need to know *how many* steps to take
- We want enough steps to take us away from our initial point



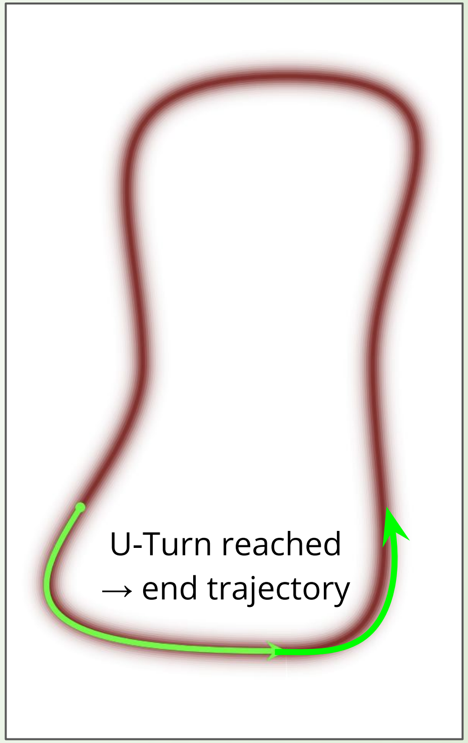
No U-Turn Sampler (NUTS)

- As well as a step-size, we need to know *how many* steps to take
- We want enough steps to take us away from our initial point
- But not so many that we return there



No U-Turn Sampler (NUTS)

- As well as a step-size, we need to know *how many* steps to take
- We want enough steps to take us away from our initial point
- But not so many that we return there
- Solution:
    - Implement a criteria that the trajectories should not perform a “U-Turn”
    - i.e. once they face the opposite direction, stop
    - NUTS was an important ingredient to make HMC work in practice for a wide-variety of problems

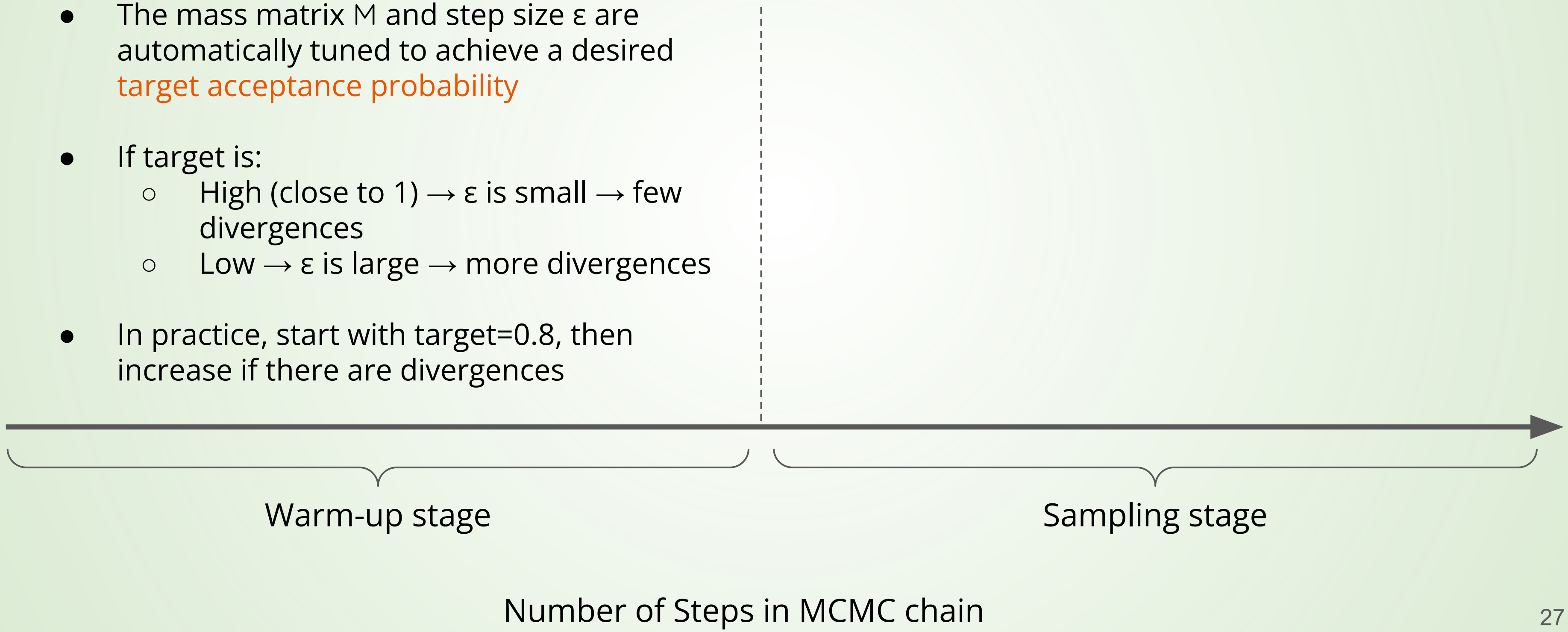


Running HMC/NUTS in practice

- The mass matrix M and step size ε are automatically tuned to achieve a desired target acceptance probability

- If target is:
  - High (close to 1) → ε is small → few divergences
  - Low → ε is large → more divergences

- In practice, start with target=0.8, then increase if there are divergences



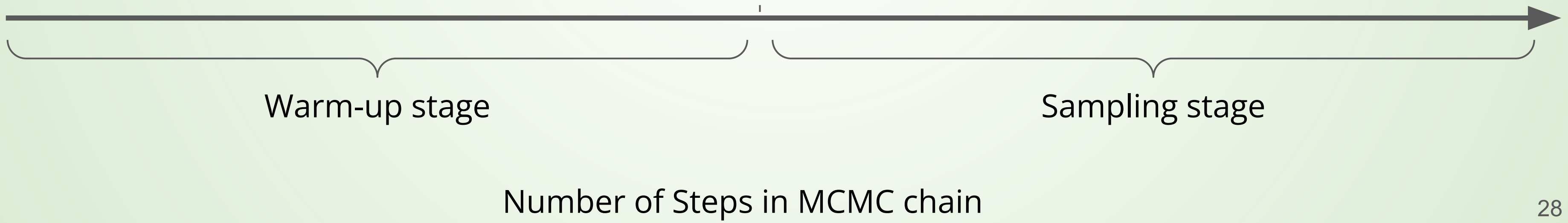
Running HMC/NUTS in practice

- The mass matrix M and step size ε are automatically tuned to achieve a desired target acceptance probability

- If target is:
  - High (close to 1) → ε is small → few divergences
  - Low → ε is large → more divergences

- In practice, start with target=0.8, then increase if there are divergences

- Sampling with HMC/NUTS
  - Uses the tuned mass matrix M and step size ε from warm-up
  - Only samples from this stage are used for inference

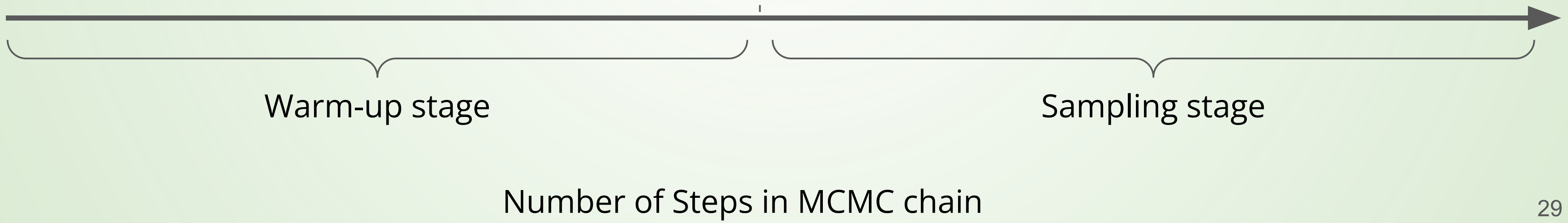


Running HMC/NUTS in practice

- The mass matrix M and step size ε are

- The length of the warm-up stage:
  - Longer warm-up allows the chain to explore the posterior geometry more fully
  - Too short may lead to a mass matrix M and step size ε which is tuned too specifically to one region of the posterior, and does not generalise well to other regions
  - Useful strategy if inference fails: increase the length of the warm up stage

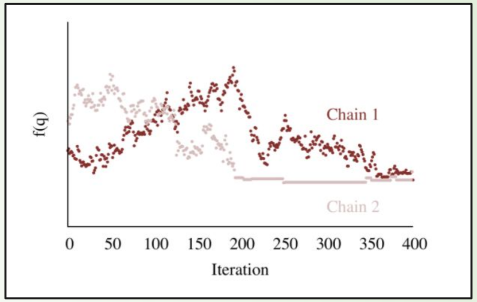
increase if there are divergences



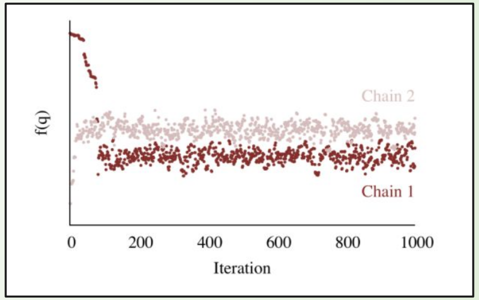
MCMC diagnostics

- Gelman Rubin statistic AKA *Rhat*
  - Has sampling converged onto a stable distribution?
  - If several chains are run (recommended!)
    - measures how well mixed different chains are
  - If one chain is run:
    - measures if first half of the chain well-mixed with the second
  - Recommended: Rhat < 1.05 for all parameters

Poorly mixed chains with Rhat >> 1



Chains not well-mixed with themselves or each-other



Chains are well-mixed internally, but not mixed or each-other

MCMC diagnostics

- Gelman Rubin statistic AKA *Rhat*
  - Has sampling converged onto a stable distribution?
  - If several chains are run (recommended!)
    - measures how well mixed different chains are
  - If one chain is run:
    - measures if first half of the chain well-mixed with the second
  - Recommended: Rhat < 1.05 for all parameters

- Effective sample size (ESS)
  - Estimate of the number of *independent* samples i.e. accounting for any autocorrelation in the chain
  - Recommended: ESS > 100 for all parameters

MCMC diagnostics

- Gelman Rubin statistic AKA *Rhat*
    - Has sampling converged onto a stable distribution?
    - If several chains are run (recommended!)
        - measures how well mixed different chains are
    - If one chain is run:
        - measures if first half of the chain well-mixed with the second
    - Recommended: Rhat < 1.05 for all parameters

- Effective sample size (ESS)
    - Estimate of the number of *independent* samples i.e. accounting for any autocorrelation in the chain
    - Recommended: ESS > 100 for all parameters

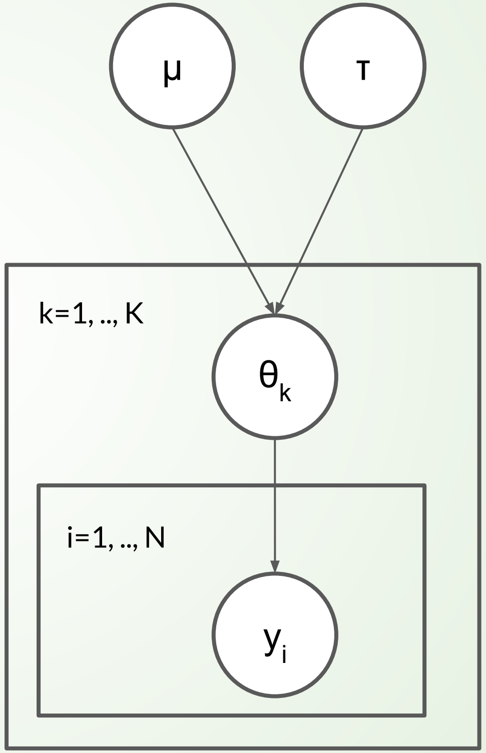
- Divergences:
    - N = 0

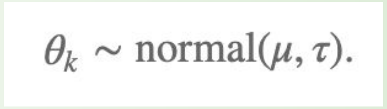
An example of problematic posterior geometry: the funnel degeneracy

Reminder: Hierarchical Models

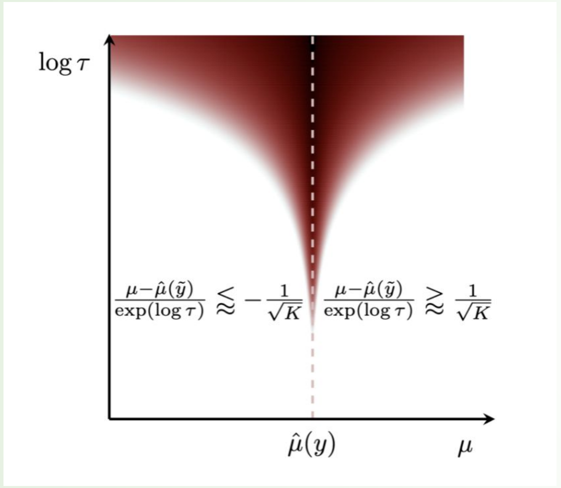
- Reminder of hierarchical models:
  - data *y* collected in *K* different *contexts*
  - different contexts have different parameters θ_k
  - *Population parameters* mean μ and scale τ control distribution of θ_k

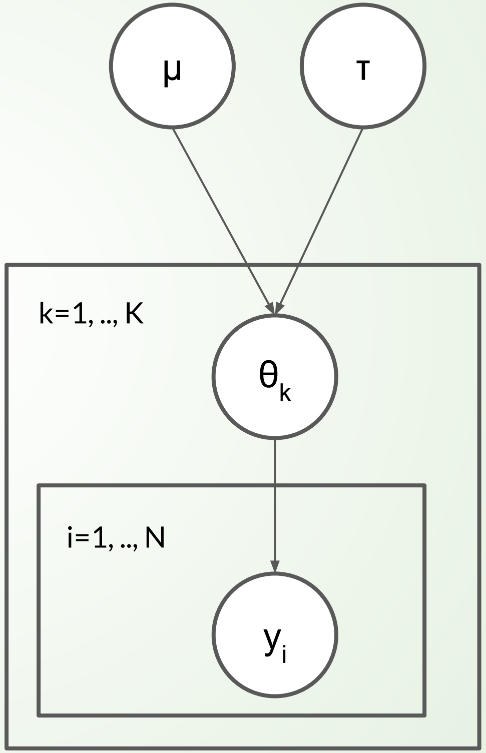
- Limits:
  - $\tau \to 0$: no variation allowed between contexts
  - $\tau \to$ infty : large variations allowed

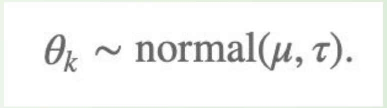



$\theta_k \sim \text{normal}(\mu, \tau).$

Funnel degeneracies in Hierarchical Models



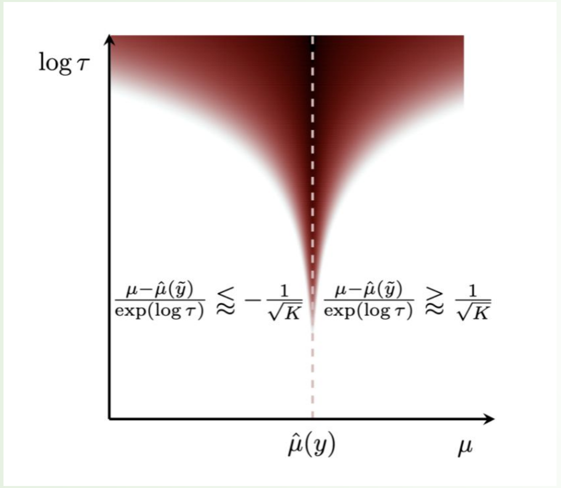




$\theta_k \sim \text{normal}(\mu, \tau).$

- Variation of τ controls the amount of variability between contexts
- Creates a funnel degeneracy with large curvature
- No single step-size will work well for narrow bottom and the wide top

Funnel degeneracies in Hierarchical Models


$\log \tau$
$\frac{\mu-\hat{\mu}(\tilde{y})}{\exp(\log \tau)} \lessapprox -\frac{1}{\sqrt{K}}$
$\frac{\mu-\hat{\mu}(\tilde{y})}{\exp(\log \tau)} \gtrapprox \frac{1}{\sqrt{K}}$
$\hat{\mu}(y)$
μ

- Variation of τ controls the amount of variability between contexts
- Creates a funnel degeneracy with large curvature
- No single step-size will work well for narrow bottom and the wide top

Solutions:

1. Collect/use stronger prior information on the population scale τ to limit the range of the funnel probed during sampling

2. Try the *non-centered parameterisation* for the per-context parameters θ_k

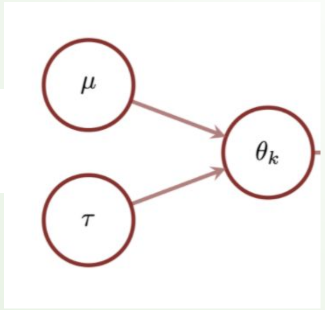
Notebook: lec3_funnel_degeneracy

Two Parameterizations of the Normal Distribution

Centered Parameterization

sample directly from the normal distribution of interest

$$\theta_k \sim \text{normal}(\mu, \tau).$$

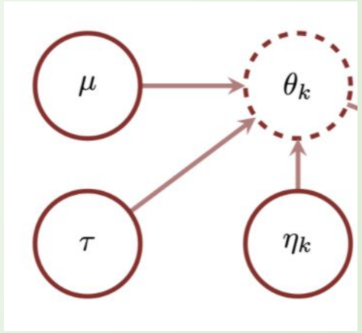


Non Centered Parameterization

sample η from unit normal
then shift by μ and re-scale by τ

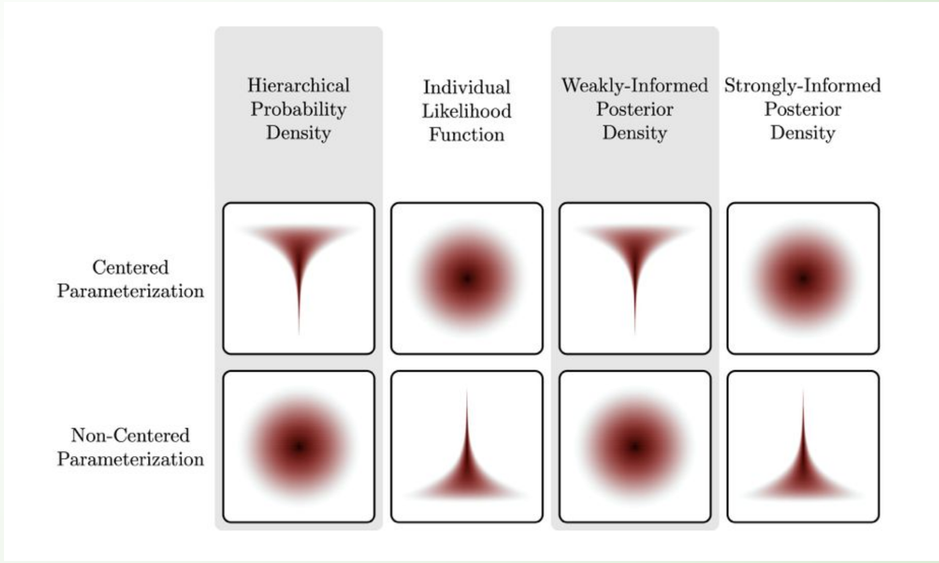
η are normalised deviations of the per-context parameters from the population parameters

$$\eta_k \sim \text{normal}(0, 1)$$
$$\theta_k = \mu + \tau \cdot \eta_k.$$



Why does the centered vs non-centred parameterisation make a difference?

- They represent exactly the same underlying model, so why does a re-parametrisation change behaviour?
- The two parameterisations have posterior geometries which are pathological in complementary regimes
- The regimes depend on the constraining power of the data:
  - τ strongly-informed by data → centered best
  - τ weakly-informed by data → non-centered best
- Let's walk-through Michael Betancourt's blogpost for more insight



Hierarchical Probability Density
Individual Likelihood Function
Weakly-Informed Posterior Density
Strongly-Informed Posterior Density
Centered Parameterization
Non-Centered Parameterization

Why does the centered vs non-centred parameterisation make a difference?

- Centered parameterisation:
    - The prior on context-dependent parameters θ_k and population scale τ displays a funnel
    - If data constrains τ, then funnel pathology is avoided in the posterior
    - When does data constrain τ?
        - enough individual contexts where θ_k well constrained
        - remaining contexts constrained by partial pooling

- Non-centered:
    - The prior on context-dependent deviation parameters η_k and population scale τ is regular (i.e. no funnel)
    - The *likelihood* now displays a funnel, since $\theta_k = (\eta_k - \mu) / \tau$
    - If data very-constraining, strong funnel is passed from likelihood to posterior
    - If data is weakly constraining, we avoid problematic posterior geometry

Notebook: lec3_funnel_degeneracy

Practical tips for running HMC/NUTS

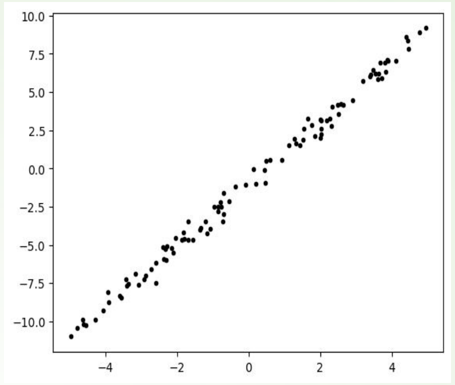
- Strategies for dealing with divergent transitions/large Rhat:
  - Use stronger prior information if available
  - Increase the target_acceptance_probability (e.g. from 0.8 to 0.9 to 0.99)
  - Increase the length of the warm-up stage (e.g. from 500 to 1000 to 2000)
  - For hierarchical models displaying a funnel degeneracy, try the non-centered parameterisations

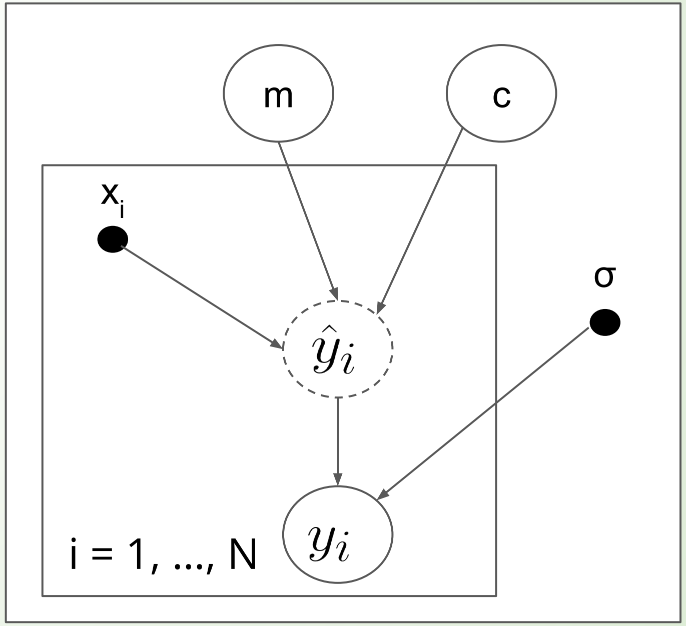
- If effective sample size is low (ESS<100), but other diagnostics look good (Rhat < 1.05 and 0 divergences):
  - increase the length of the sampling stage

Gaussian Processes

Motivation

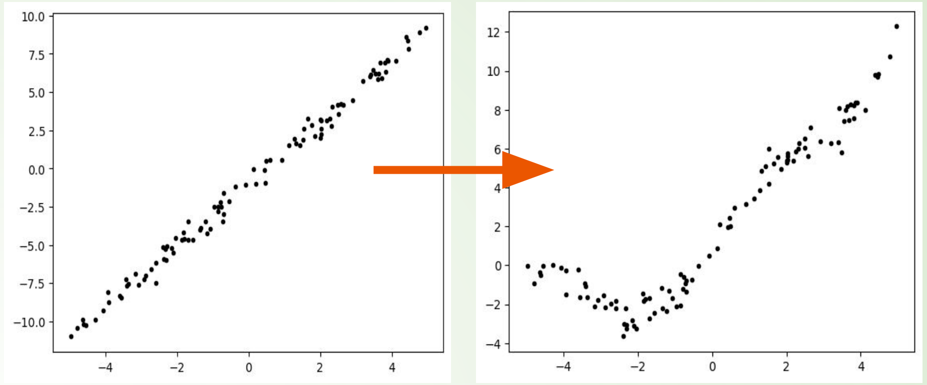
- Linear regression:
  - Two unknown parameters
  - slope m, and intercept c

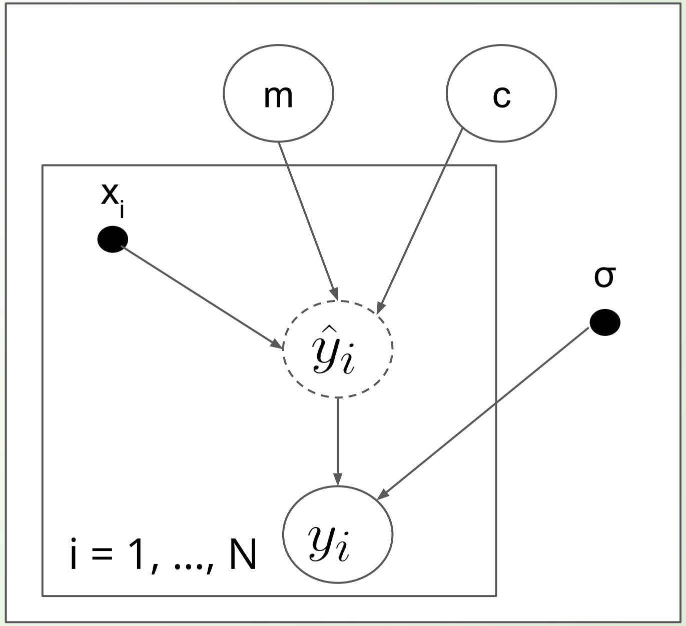




Motivation

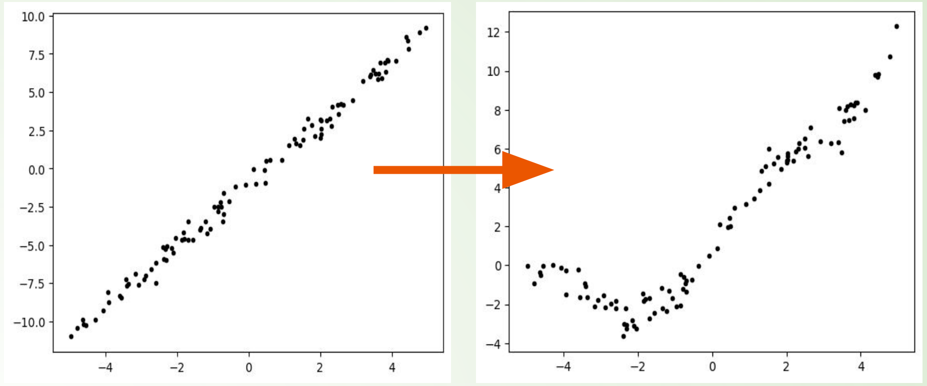
- Linear regression:
  - Two unknown parameters
  - slope m, and intercept c
- What if the data is more complicated than a linear relation?

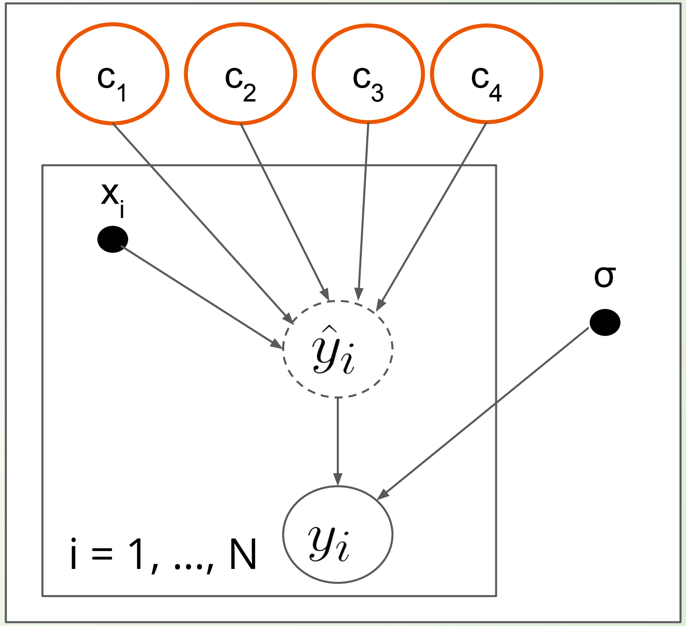




Motivation

- Linear regression:
  - Two unknown parameters
  - slope m, and intercept c
- What if the data is more complicated than a linear relation?
- Option 1: use a more complex parameterised function
  - e.g. order-N polynomial, with coefficients c_i
  - but we don't know if choice of parameterisation is suitable

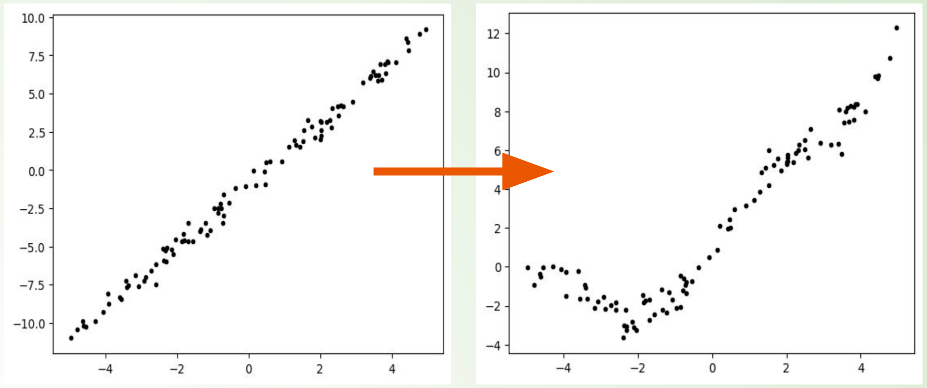


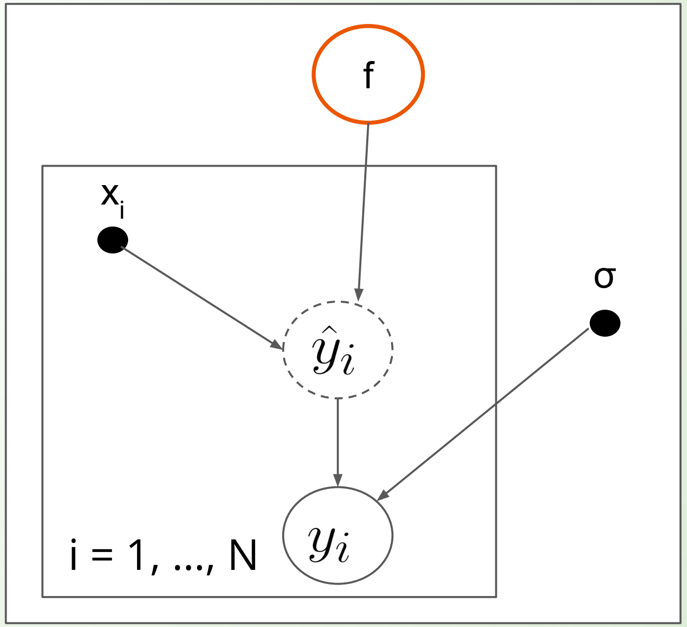


Motivation

- Linear regression:
  - Two unknown parameters
  - slope m, and intercept c
- What if the data is more complicated than a linear relation?
- Option 1: use a more complex parameterised function
  - e.g. order-N polynomial, with coefficients c_i
  - but we don't know if choice of parameterisation is suitable
- Option 2: directly infer a non-parametric function f which describes the data

  i.e. y = f(x)

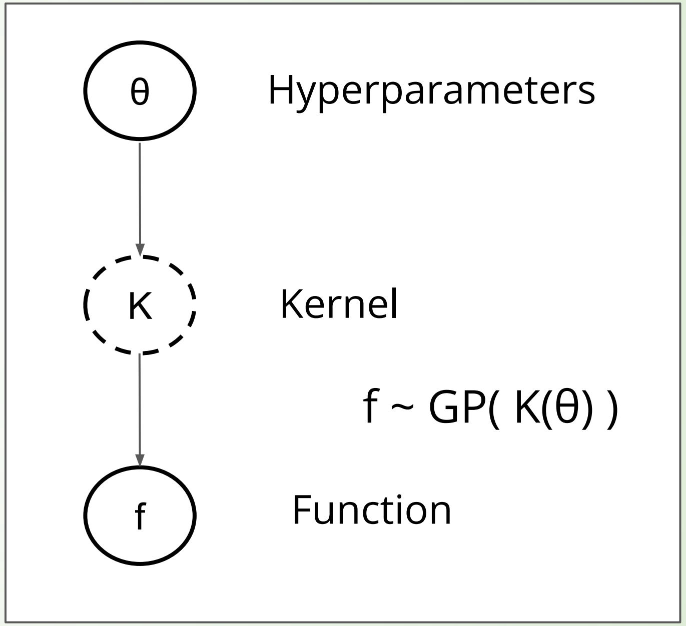




Gaussian Processes (GPs)

- GPs are priors over function spaces
- We can sample *functions* from a Gaussian Process prior
- The GP depends on a kernel K
- The kernel K depends on hyperparameters θ
- Sampling notation:

  $f \sim GP(K(\theta))$



The Gaussian Process Kernel

- The kernel K controls the characteristics of the function

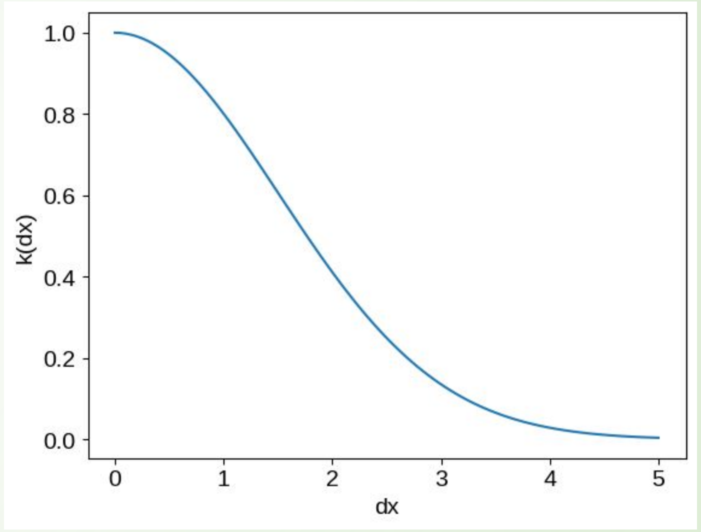
- For two points (x_1 , x_2) in the input space with separation dx = $|x_1 - x_2|$

  K(dx)

  determines how close $f(x_1)$ and $f(x_2)$ are

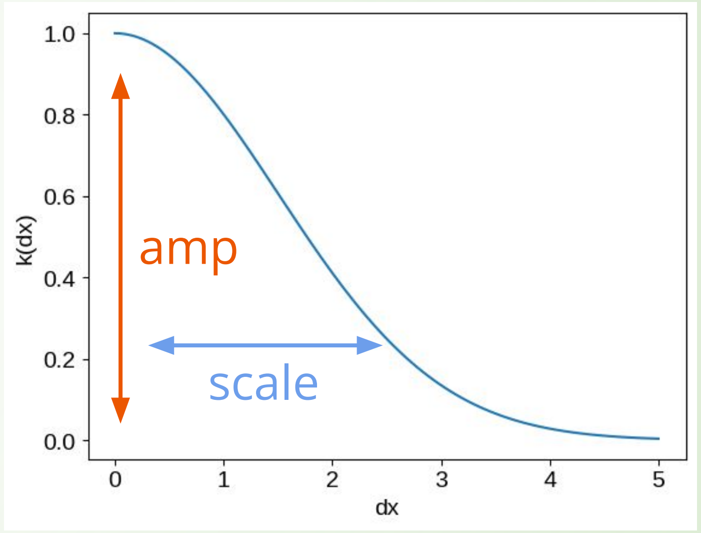
- i.e. the *smoothness* of f

- Kernel controlled by two hyperparameters:
  - Lengthscale
  - Amplitude



The Gaussian Process Kernel

- Amplitude:
  - Larger amplitude → the function can have larger vertical variations
- Lengthscale:
  - Smaller lengthscale → the function can have more high-frequency variations



Notebook:
lec3_gaussian_processes

Gaussian Process: extensions

- We've looked at simple example: 1D regression
- The `tinygp` documentation contains many more examples:
    - Using more complex kernels e.g. to model periodic functions
    - Using GPs to model deviations from a parametric mean function
    - Using GPs within more complicated `numpyro` models