Probabilistic Programming Languages

Guillaume Baudart
Christine Tasson

MPRI 2021-2022

MCMC Metropolis-Hastings

Probabilistic Programming Languages

Markov Chain Monte Carlo (MCMC)

Main idea

- Create a Markov chain that converge to the posterior distribution
- Iterate the process until convergence
- Generate samples to approximate the distribution

Pros

- Faster convergence
- Better results for high-dimensional models
- Advanced state-of-the-art optimizations (e.g., HMC, NUTS).

Cons

- Convergences?
- Traps: multimodal, funnel
- Samples correlation

Reminder: Rejection Sampling

coin.ml

```
let coin prob data =
  let z = sample prob (uniform ~a:0. ~b:1.) in
  List.iter (observe prob (bernoulli ~p:z)) data;
  z

let _ =
  let d = infer coin [ 1; 1; 0; 0; 0; 0; 0; 0; 0; 0 ] in
  plot d
```

Executing the model generates one sample

- `sample`: draw from a distribution
- `assume`/`observe`: hard conditioning, reject invalid samples
- Terminates with n valid samples

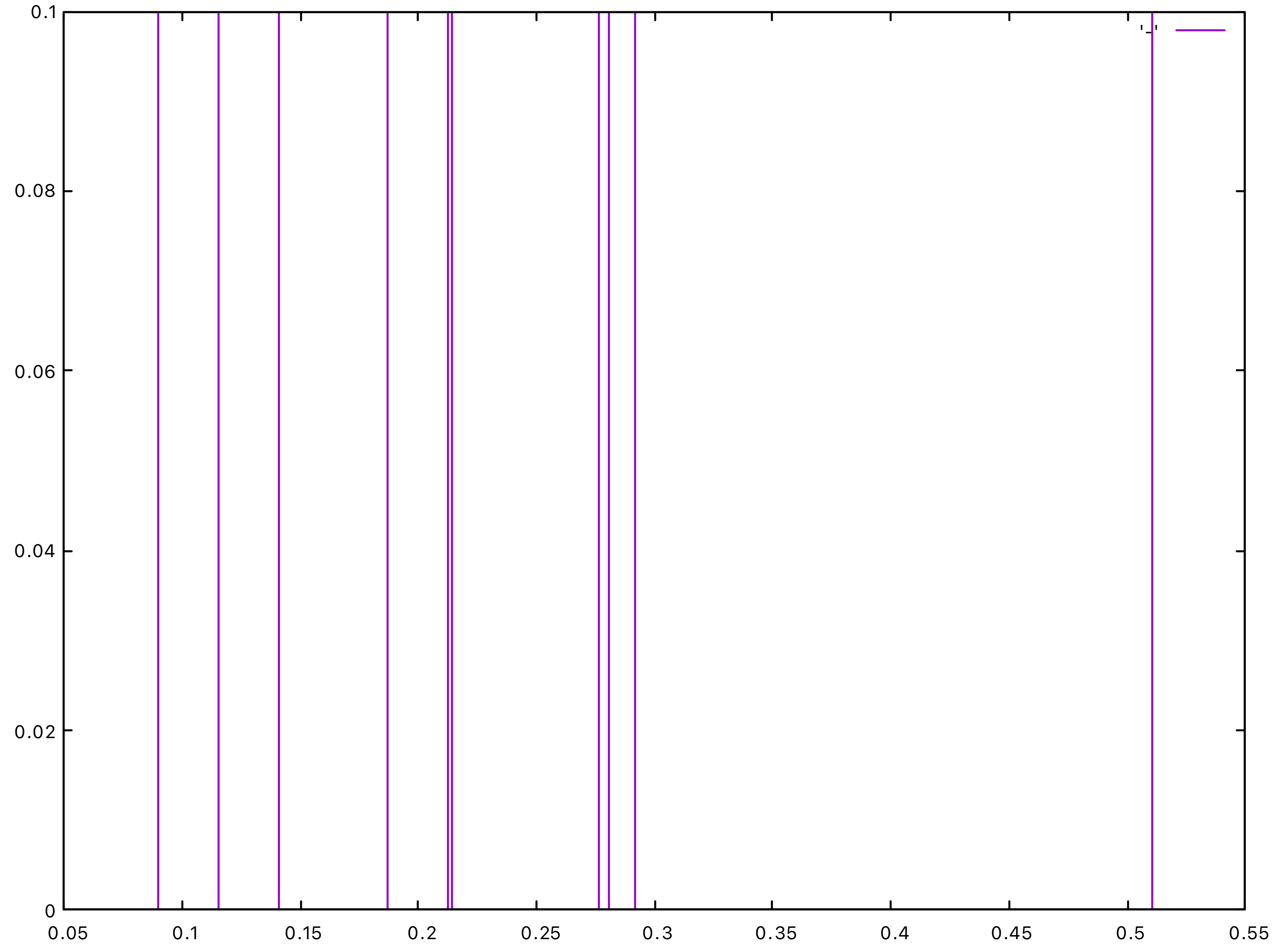
Reminder: Rejection Sampling

coin.ml

```
let coin prob data =
  let z = sample prob (uniform ~a:0. ~b:1.) in
  List.iter (observe prob (bernoulli ~p:z)) data;
  z

let _ =
  let d = infer coin [ 1; 1; 0; 0; 0; 0; 0; 0; 0; 0 ] in
  plot d
```

10 particles



Executing the model generates one sample

- `sample`: draw from a distribution
- `assume`/`observe`: hard conditioning, reject invalid samples
- Terminates with n valid samples

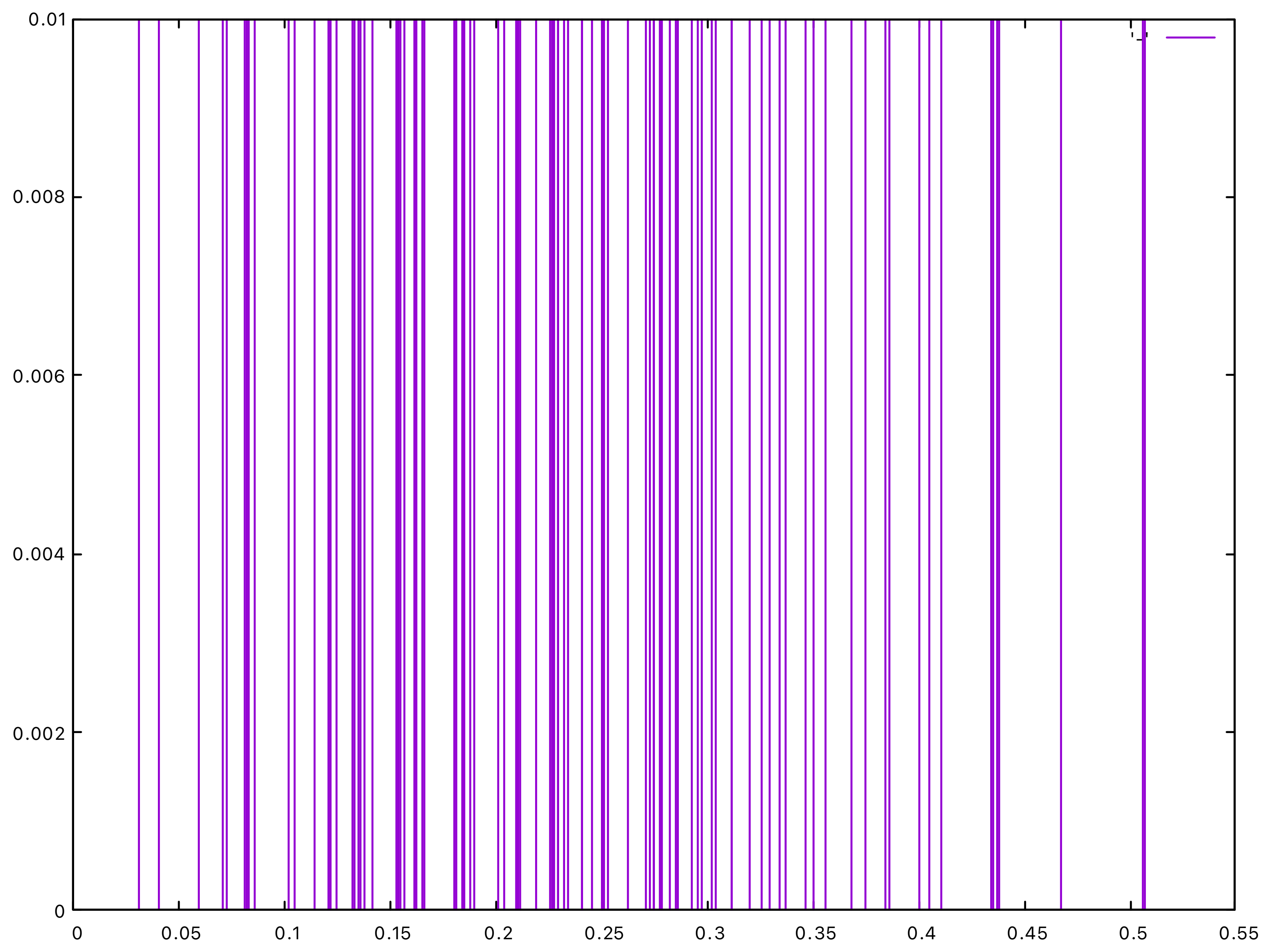
Reminder: Rejection Sampling

coin.ml

```
let coin prob data =
  let z = sample prob (uniform ~a:0. ~b:1.) in
  List.iter (observe prob (bernoulli ~p:z)) data;
  z

let _ =
  let d = infer coin [ 1; 1; 0; 0; 0; 0; 0; 0; 0; 0 ] in
  plot d
```

100 particles



Executing the model generates one sample

- `sample`: draw from a distribution
- `assume`/`observe`: hard conditioning, reject invalid samples
- Terminates with n valid samples

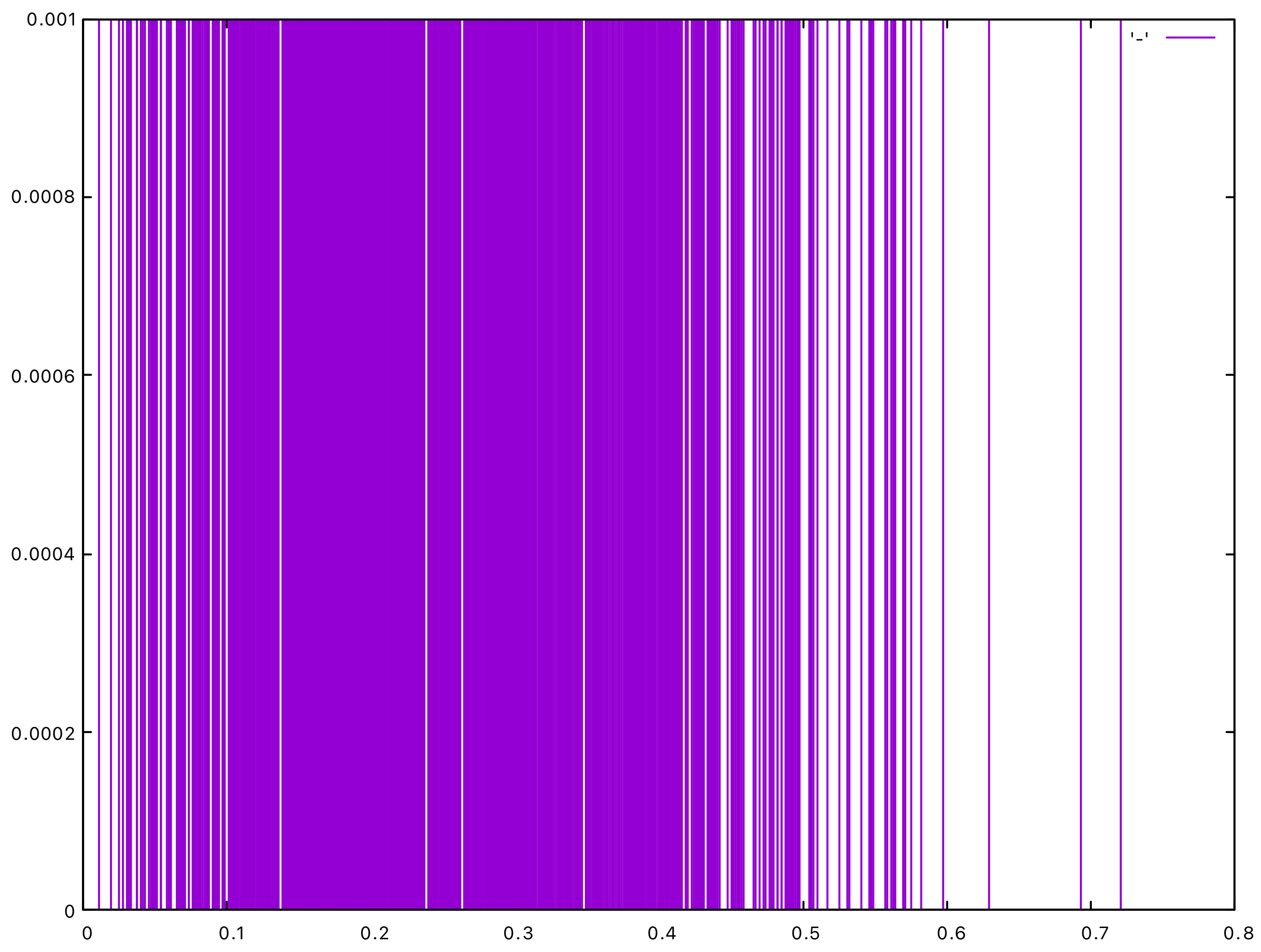
Reminder: Rejection Sampling

coin.ml

```
let coin prob data =
  let z = sample prob (uniform ~a:0. ~b:1.) in
  List.iter (observe prob (bernoulli ~p:z)) data;
  z

let _ =
  let d = infer coin [ 1; 1; 0; 0; 0; 0; 0; 0; 0; 0 ] in
  plot d
```

1000 particles



Executing the model generates one sample

- `sample`: draw from a distribution
- `assume`/`observe`: hard conditioning, reject invalid samples
- Terminates with n valid samples

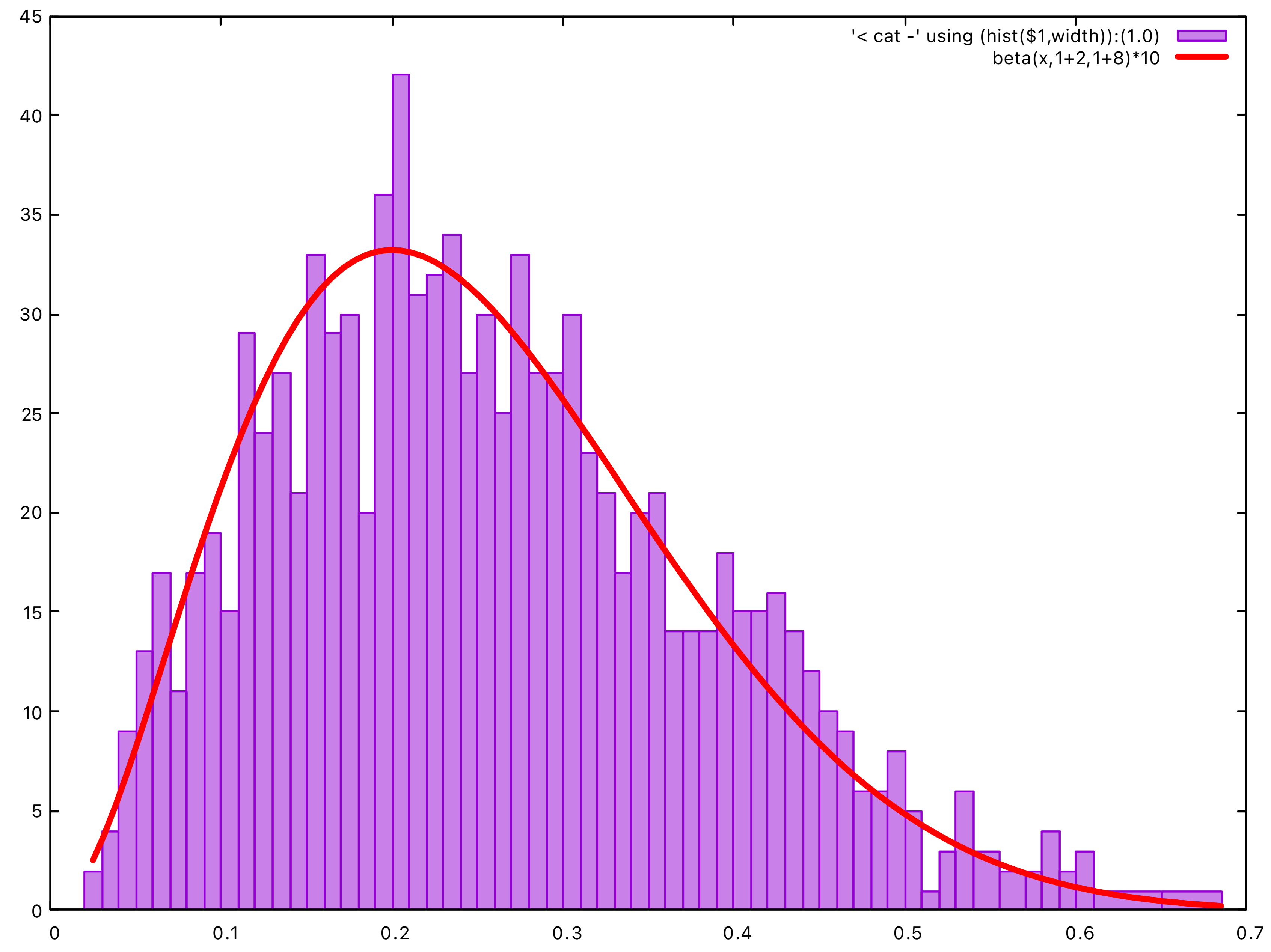
Reminder: Rejection Sampling

coin.ml

```
let coin prob data =
  let z = sample prob (uniform ~a:0. ~b:1.) in
  List.iter (observe prob (bernoulli ~p:z)) data;
  z

let _ =
  let d = infer coin [ 1; 1; 0; 0; 0; 0; 0; 0; 0; 0 ] in
  plot d
```

1000 particles



Executing the model generates one sample

- `sample`: draw from a distribution
- `assume`/`observe`: hard conditioning, reject invalid samples
- Terminates with n valid samples

Weighted Rejection Sampling

Adapt rejection sampling to soft conditioning

- Execute the sampler to get a pair (v_i, w_i)
- Suppose $w_{\max}$ is known
- Accept the sample with probability $w_i / w_{\max}$ or retry

Weighted Rejection Sampling

Adapt rejection sampling to soft conditioning

- Execute the sampler to get a pair (v_i, w_i)
- Suppose $w_{\max}$ is known
- Accept the sample with probability $w_i / w_{\max}$ or retry

But $w_{\max}$ is not known...

Execution Trace

Consider a program execution with

- $X = x_0, \ldots, x_n$: set of random variables sampled at step i : the trace
- $Y = y_0, \ldots, y_m$: set of random variables observed at step i.

Remarks

- Sets X and Y depend on the execution path
- We can only control X

Execution Trace

Consider a program execution with

- $X = x_0, \ldots, x_n$: set of random variables sampled at step i : the trace
- $Y = y_0, \ldots, y_m$: set of random variables observed at step i.

Remarks

- Sets X and Y depend on the execution path
- We can only control X

```
let bimodal y =
  let z = sample (bernoulli ~p:0.5) in
  let mu =
    if z then sample (gaussian ~mu:-1. ~sigma:1.)
    else sample (gaussian ~mu:1. ~sigma:1.)
  in
  let () = observe (gaussian ~mu ~sigma:1.) y in
  z
```

Multi-Sites Metropolis Hastings

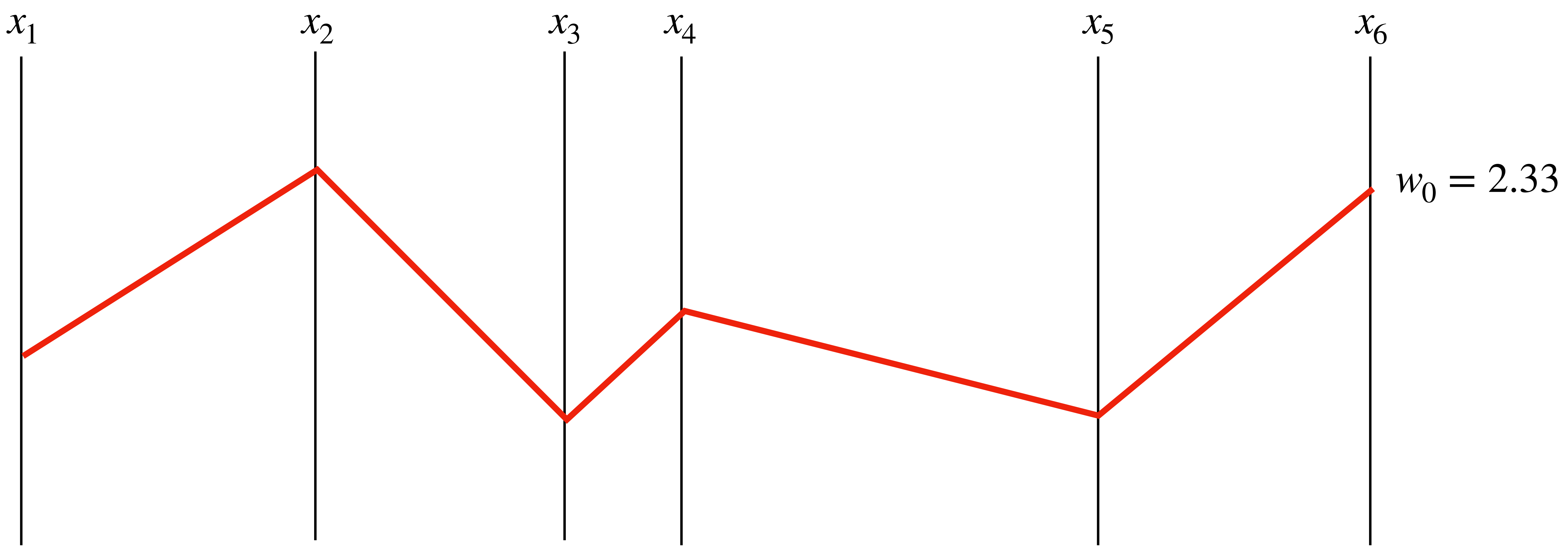
Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i / w_{i-1}
- Otherwise return the previous trace X_{i-1}

Multi-Sites Metropolis Hastings

Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i / w_{i-1}
- Otherwise return the previous trace X_{i-1}



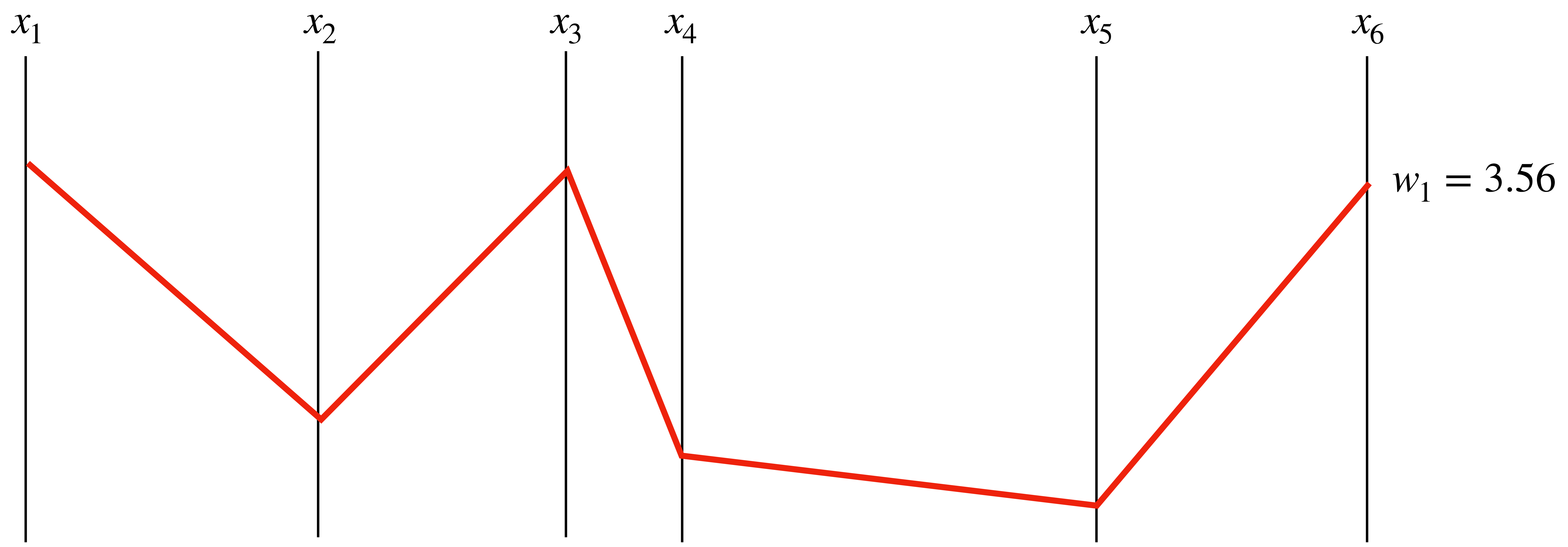
outputs

v_0

Multi-Sites Metropolis Hastings

Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i / w_{i-1}
- Otherwise return the previous trace X_{i-1}



outputs

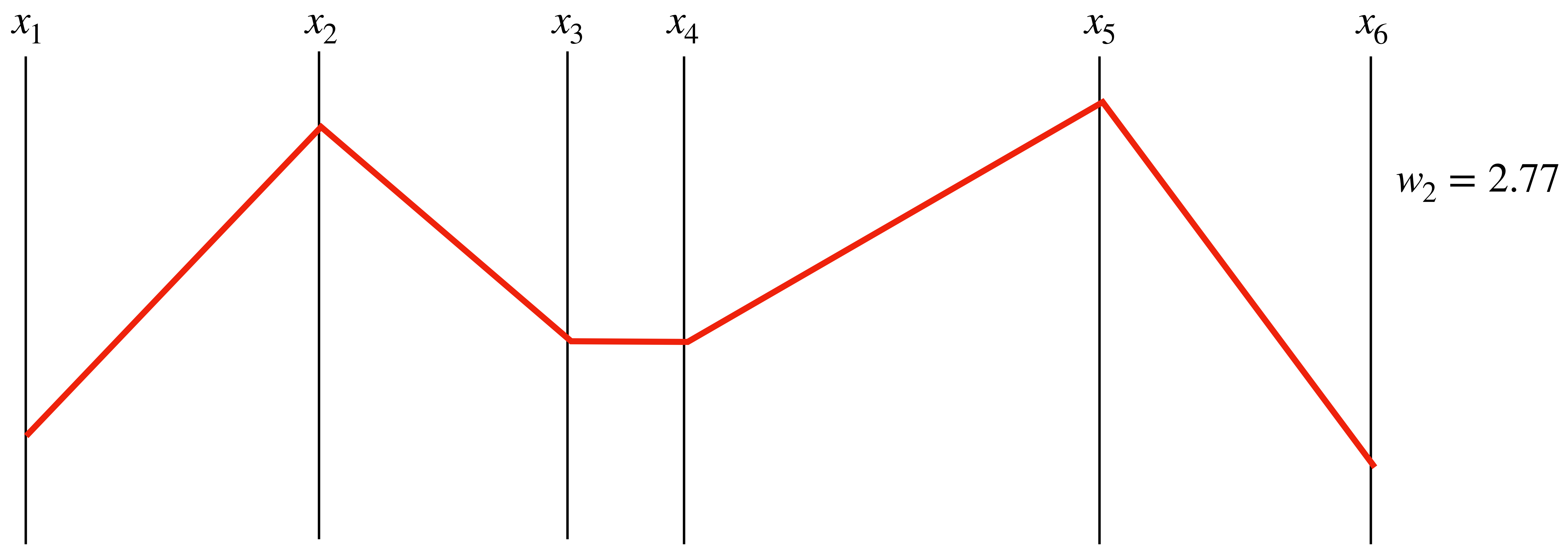
v_0

v_1

Multi-Sites Metropolis Hastings

Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i/w_{i-1}
- Otherwise return the previous trace X_{i-1}



outputs

v_0

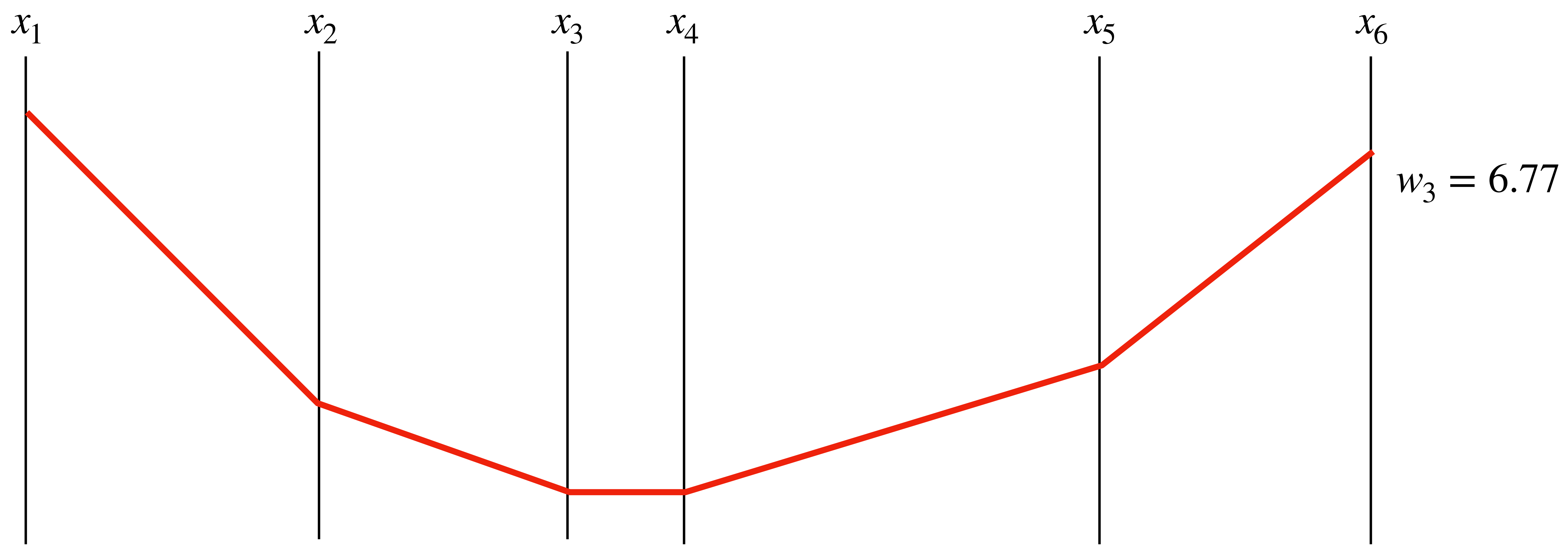
v_1

v_1

Multi-Sites Metropolis Hastings

Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i / w_{i-1}
- Otherwise return the previous trace X_{i-1}



outputs

v_0

v_1

v_1

v_3

...

General Metropolis Hastings

More generally

- $X = x_0, \ldots, x_n$: set of random variables sampled at step i: the trace
- $Y = y_0, \ldots, y_m$: set of random variables observed at step i.
- Propose a new trace from a proposal distribution $q(X_i | X_{i-1})$
- Accept the trace with probability α, where

$$\alpha = \min\left(1, \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})}\right)$$

- Otherwise return the previous trace X_{i-1}

Multi-Sites Metropolis Hastings: Acceptation

- Draw proposal from priors: $q(X_i \mid X_{i-1}) = p(X_i)$
- Resample all variable in X_i at each step

$$
\begin{aligned}
\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} &= \frac{p(Y_i | X_i)\, p(X_i)}{p(Y_{i-1} \mid X_{i-1})\, p(X_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} \\
&= \frac{p(Y_i | X_i)\, p(X_i)}{p(Y_{i-1} \mid X_{i-1})\, p(X_{i-1})} \frac{p(X_{i-1})}{p(X_i)} \\
&= \frac{p(Y_i | X_i)}{p(Y_{i-1} \mid X_{i-1}))} \\
&= \frac{w_i}{w_{i-1}}
\end{aligned}
$$

Multi-Sites Metropolis Hastings: Acceptation

- Draw proposal from priors: $q(X_i \mid X_{i-1}) = p(X_i)$
- Resample all variable in X_i at each step

$$
\begin{aligned}
\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} &= \frac{p(Y_i | X_i)\ p(X_i)}{p(Y_{i-1} \mid X_{i-1})\ p(X_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} \\
&= \frac{p(Y_i | X_i)\ p(X_i)}{p(Y_{i-1} \mid X_{i-1})\ p(X_{i-1})} \frac{p(X_{i-1})}{p(X_i)} \\
&= \frac{p(Y_i | X_i)}{p(Y_{i-1} \mid X_{i-1}))} \\
&= \frac{w_i}{w_{i-1}}
\end{aligned}
$$

Markov chain on execution traces

- Execute the sampler to get a trace and associated score (X_i, w_i)
- If $w_i \geq w_{i-1}$ accept the trace (and the associated output)
- Else accept the trace with probability w_i / w_{i-1}
- Otherwise return the previous trace X_{i-1}

Single-Site Metropolis Hastings

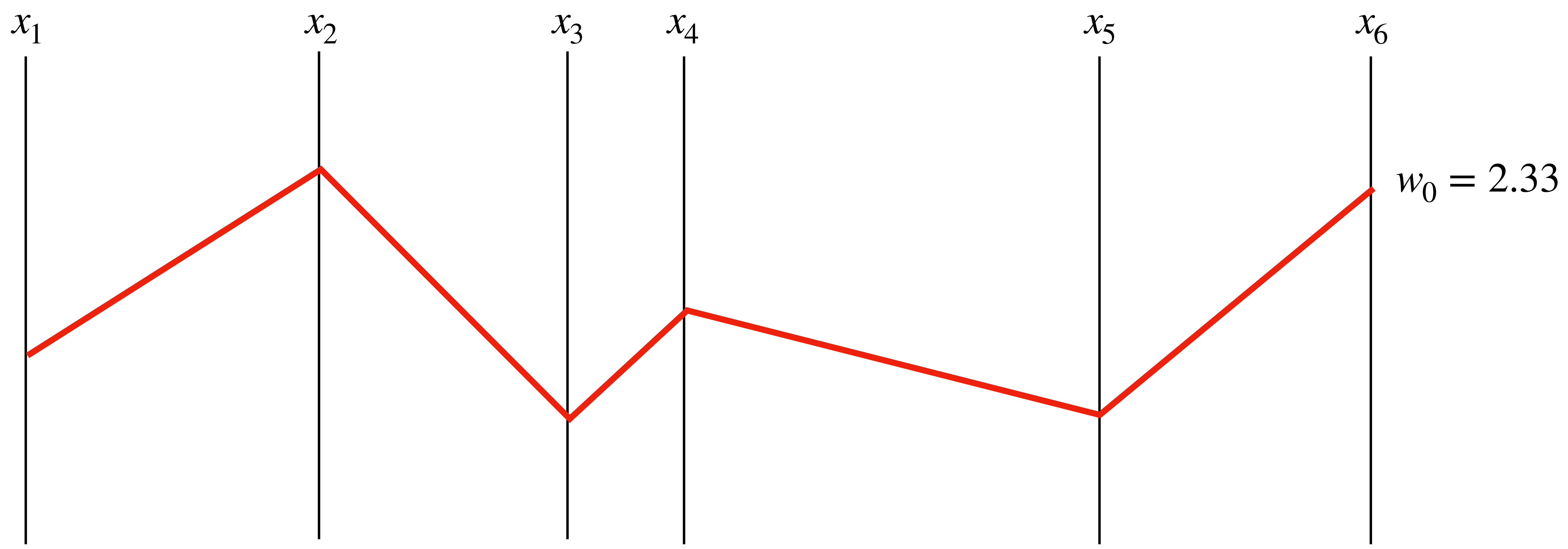
Reuse most of the previous trace (i.e., sampled values)

- Choose one random variable to resample to obtain a new execution
- Accept the trace with probability α
- Otherwise use the previous trace

Single-Site Metropolis Hastings

Reuse most of the previous trace (i.e., sampled values)

- Choose one random variable to resample to obtain a new execution
- Accept the trace with probability α
- Otherwise use the previous trace



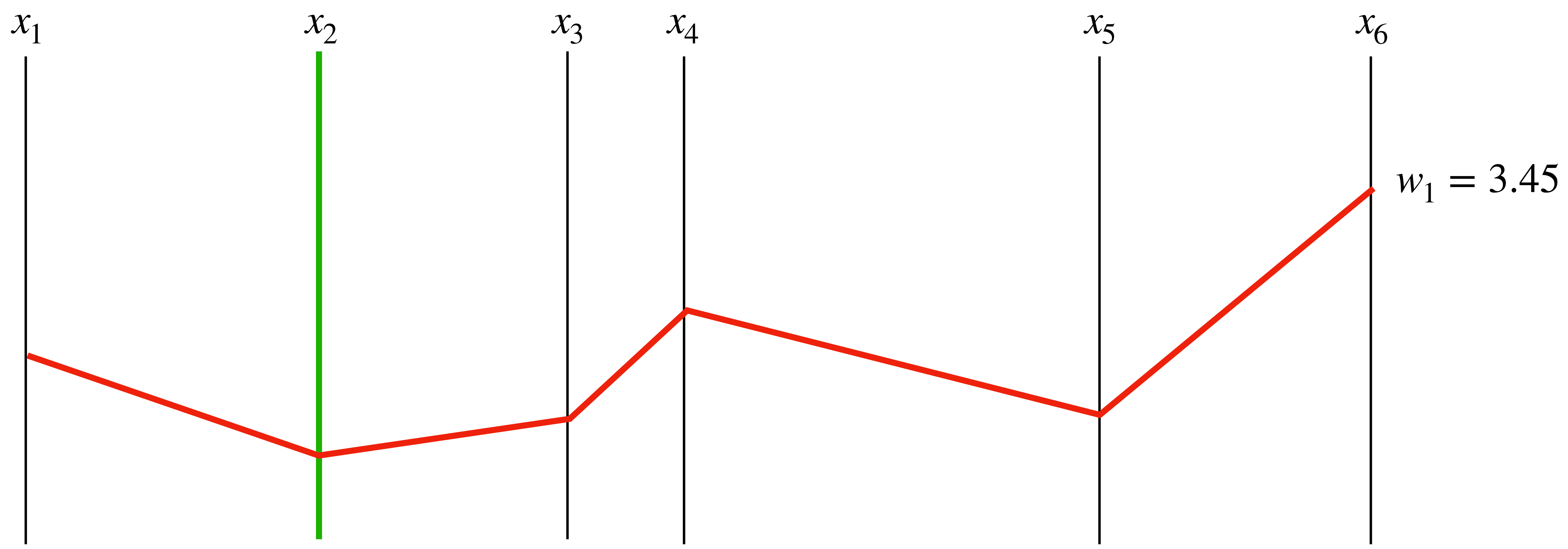
samples

v_0

Single-Site Metropolis Hastings

Reuse most of the previous trace (i.e., sampled values)

- Choose one random variable to resample to obtain a new execution
- Accept the trace with probability α
- Otherwise use the previous trace



samples

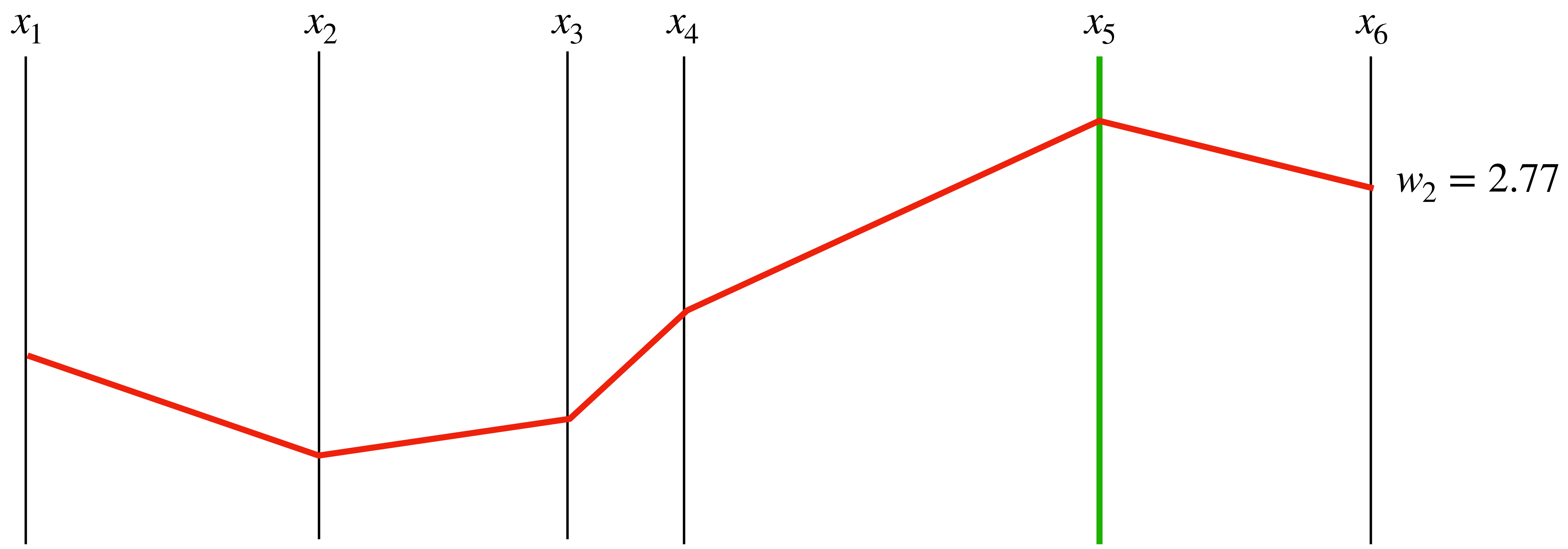
v_0

v_1

Single-Site Metropolis Hastings

Reuse most of the previous trace (i.e., sampled values)

- Choose one random variable to resample to obtain a new execution
- Accept the trace with probability α
- Otherwise use the previous trace

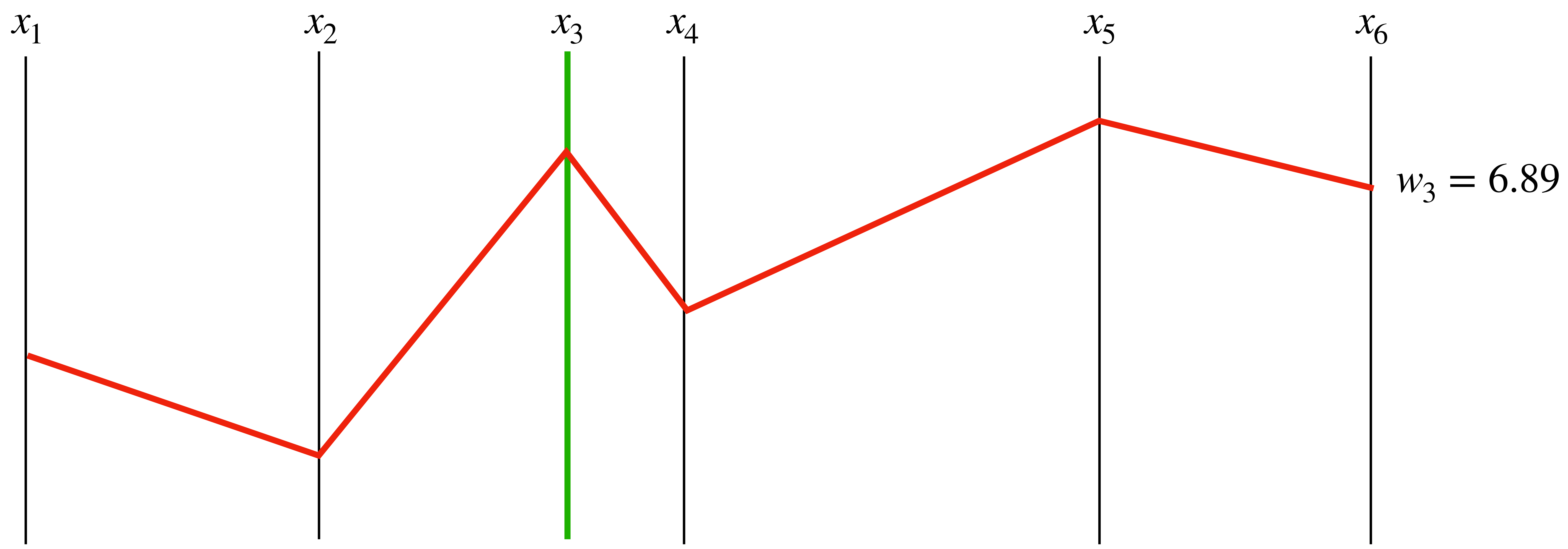


samples
v_0
v_1
v_1

Single-Site Metropolis Hastings

Reuse most of the previous trace (i.e., sampled values)

- Choose one random variable to resample to obtain a new execution
- Accept the trace with probability α
- Otherwise use the previous trace



samples

v_0

v_1

v_1

v_3

Single Site Metropolis Hastings: Acceptation

Track the likelihood of all random variable during execution

- `x = sample d → w(x) = (x, pdf d x)`
- `observe d y  → w(y) = (y, pdf d y)`

Single Site Metropolis Hastings: Acceptation

Track the likelihood of all random variable during execution

- `x = sample d → w(x) = (x, pdf d x)`
- `observe d y  → w(y) = (y, pdf d y)`

```
let bimodal y =
  let z = sample (bernoulli ~p:0.5) in
  let mu =
    if z then sample (gaussian ~mu:-1. ~sigma:1.)
    else sample (gaussian ~mu:1. ~sigma:1.)
  in
  let () = observe (gaussian ~mu ~sigma:1.) y in
  z
```

Single Site Metropolis Hastings: Acceptation

Track the likelihood of all random variable during execution

- `x = sample d → w(x) = (x, pdf d x)`
- `observe d y  → w(y) = (y, pdf d y)`

```
let bimodal y =
  let z = sample (bernoulli ~p:0.5) in
  let mu =
    if z then sample (gaussian ~mu:-1. ~sigma:1.)
    else sample (gaussian ~mu:1. ~sigma:1.)
  in
  let () = observe (gaussian ~mu ~sigma:1.) y in
  z
```

```
w(z)  = (false, 0.5)
w(mu) = (1.2, 0.40)
w(y)  = (2.0, 0.27)
```

Single Site Metropolis Hastings: Acceptation

Track the likelihood of all random variable during execution

- `x = sample d → w(x) = (x, pdf d x)`
- `observe d y  → w(y) = (y, pdf d y)`   score (as in importance sampling)

```
let bimodal y =
  let z = sample (bernoulli ~p:0.5) in
  let mu =
    if z then sample (gaussian ~mu:-1. ~sigma:1.)
    else sample (gaussian ~mu:1. ~sigma:1.)
  in
  let () = observe (gaussian ~mu ~sigma:1.) y in
  z
```

```
w(z)  = (false, 0.5)
w(mu) = (1.2, 0.40)
w(y)  = (2.0, 0.27)
```

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} = \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\ q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\ q(x_0 \mid X_{i-1})}$$

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} = \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\ q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\ q(x_0 \mid X_{i-1})}$$

$$p(X, Y) = \prod_{x \in X} w(x) \prod_{y \in Y} w(y)$$

(not necessarily independent)

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} = \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\ q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\ q(x_0 \mid X_{i-1})}$$

$$p(X, Y) = \prod_{x \in X} w(x) \prod_{y \in Y} w(y)$$

(not necessarily independent)

$$q(x_0 | X) = 1/|X|$$

1/(n choices)

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} = \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\; q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\; q(x_0 \mid X_{i-1})}$$

$$p(X, Y) = \prod_{x \in X} w(x) \prod_{y \in Y} w(y)$$

(not necessarily independent)

$$q(x_0 \mid X) = 1/|X|$$

1/(n choices)

$$q(X \mid X', x_0) = \prod_{x \in X^{\text{sample}}} w(x)$$

Resampled from the prior

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$p(X, Y) = \prod_{x \in X} w(x) \prod_{y \in Y} w(y)$$

$$q(x_0 | X) = |X|$$

$$q(X | X', x_0) = \prod_{x \in X^{\text{sample}}} w(x)$$

$$\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} = \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\, q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\, q(x_0 \mid X_{i-1})}$$

$$= \frac{q(x_0 \mid X_i)}{q(x_0 \mid X_{i-1})} \frac{p(X_i, Y_i)}{q(X_i \mid X_{i-1}, x_0)} \frac{q(X_{i-1} \mid X_i, x_0)}{p(X_{i-1}, Y_{i-1})}$$

$$= \frac{|X_{i-1}|}{|X_i|} \frac{\prod_{x \in X_i} w(x) \ \prod_{y \in Y_i} w(y)}{\prod_{x \in X_i^{\text{sample}}} w(x)} \frac{\prod_{x \in X_{i-1}^{\text{sample}}} w(x)}{\prod_{x \in X_{i-1}} w(x) \ \prod_{y \in Y_{i-1}} w(y)}$$

$$= \frac{|X_{i-1}|}{|X_i|} \frac{\prod_{x \in X_i^{\text{reuse}}} w(x)}{\prod_{x \in X_{i-1}^{\text{reuse}}} w(x)} \frac{\prod_{y \in Y_i} w(y)}{\prod_{y \in Y_{i-1}} w(y)}$$

Single Site Metropolis Hastings: Acceptation

- Pick one variable x_0 at random in the trace
- Resample only this variable and all its dependencies $X = X^{\text{sample}} \cup X^{\text{reuse}}$

$$p(X, Y) = \prod_{x \in X} w(x) \prod_{y \in Y} w(y)$$

$$q(x_0 | X) = |X|$$

$$q(X | X', x_0) = \prod_{x \in X^{\text{sample}}} w(x)$$

$$\begin{aligned}
\frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i)}{q(X_i \mid X_{i-1})} &= \frac{p(X_i, Y_i)}{p(X_{i-1}, Y_{i-1})} \frac{q(X_{i-1} \mid X_i, x_0)\; q(x_0 \mid X_i)}{q(X_i \mid X_{i-1}, x_0)\; q(x_0 \mid X_{i-1})} \\
&= \frac{q(x_0 \mid X_i)}{q(x_0 \mid X_{i-1})} \frac{p(X_i, Y_i)}{q(X_i \mid X_{i-1}, x_0)} \frac{q(X_{i-1} \mid X_i, x_0)}{p(X_{i-1}, Y_{i-1})} \\
&= \frac{|X_{i-1}|}{|X_i|} \frac{\prod_{x \in X_i} w(x) \;\prod_{y \in Y_i} w(y)}{\prod_{x \in X_i^{\text{sample}}} w(x)} \frac{\prod_{x \in X_{i-1}^{\text{sample}}} w(x)}{\prod_{x \in X_{i-1}} w(x) \;\prod_{y \in Y_{i-1}} w(y)} \\
&= \underbrace{\frac{|X_{i-1}|}{|X_i|}}_{\text{choice } x_0} \underbrace{\frac{\prod_{x \in X_i^{\text{reuse}}} w(x)}{\prod_{x \in X_{i-1}^{\text{reuse}}} w(x)}}_{\text{reused}} \underbrace{\frac{\prod_{y \in Y_i} w(y)}{\prod_{y \in Y_{i-1}} w(y)}}_{\text{scores}}
\end{aligned}$$

Implementation

Implementation

Option 1

- CPS model
- Pick x_0 at random
- Resample all variables after x_0

Implementation

Option 1

- CPS model
- Pick x_0 at random
- Resample all variables after x_0

Option 2

- CPS model
- Pick x_0 at random
- Memoize values of all random variable
- Resample only when needed

Implementation

Option 1

- CPS model
- Pick x_0 at random
- Resample all variables after x_0

Option 2

- CPS model
- Pick x_0 at random
- Memoize values of all random variable
- Resample only when needed

Warning: Memoization naming scheme
Different variable may have the same name...

Limitations

Limitations

Convergence

- Theoretical conditions are complex
- Must be checked experimentally
- Diagnostic tools: trace plot, R-hat (multi-chains)
- Solution: warmup, change initial conditions, reparameterization, ...

Limitations

Convergence

- Theoretical conditions are complex
- Must be checked experimentally
- Diagnostic tools: trace plot, R-hat (multi-chains)
- Solution: warmup, change initial conditions, reparameterization, ...

Sample correlation

- Next sample depends on the previous one
- Diagnostic tools ESS (effective sample size)
- Solution: thinning, ...

Limitations

Convergence

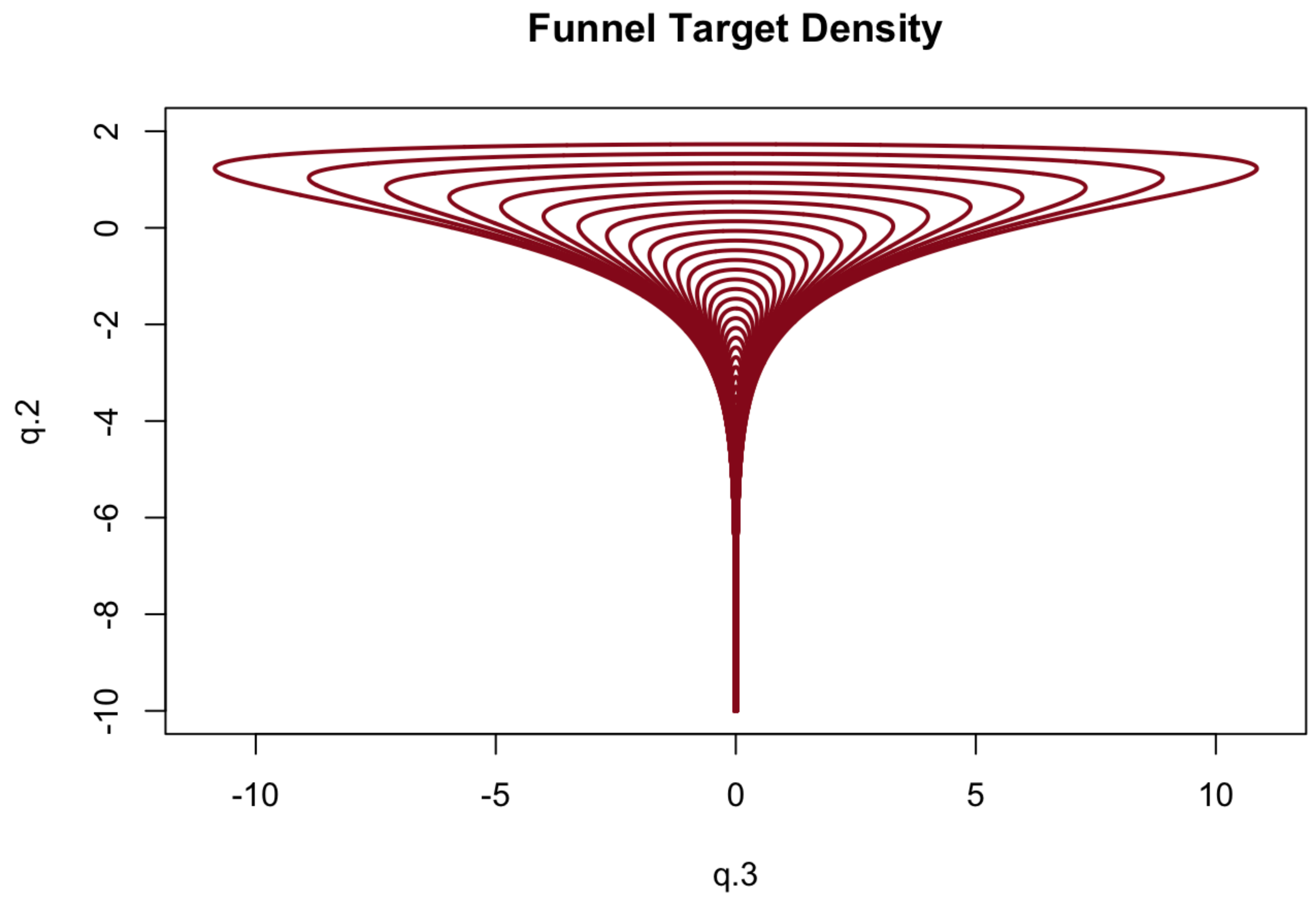
- Theoretical conditions are complex
- Must be checked experimentally
- Diagnostic tools: trace plot, R-hat (multi-chains)
- Solution: warmup, change initial conditions, reparameterization, ...

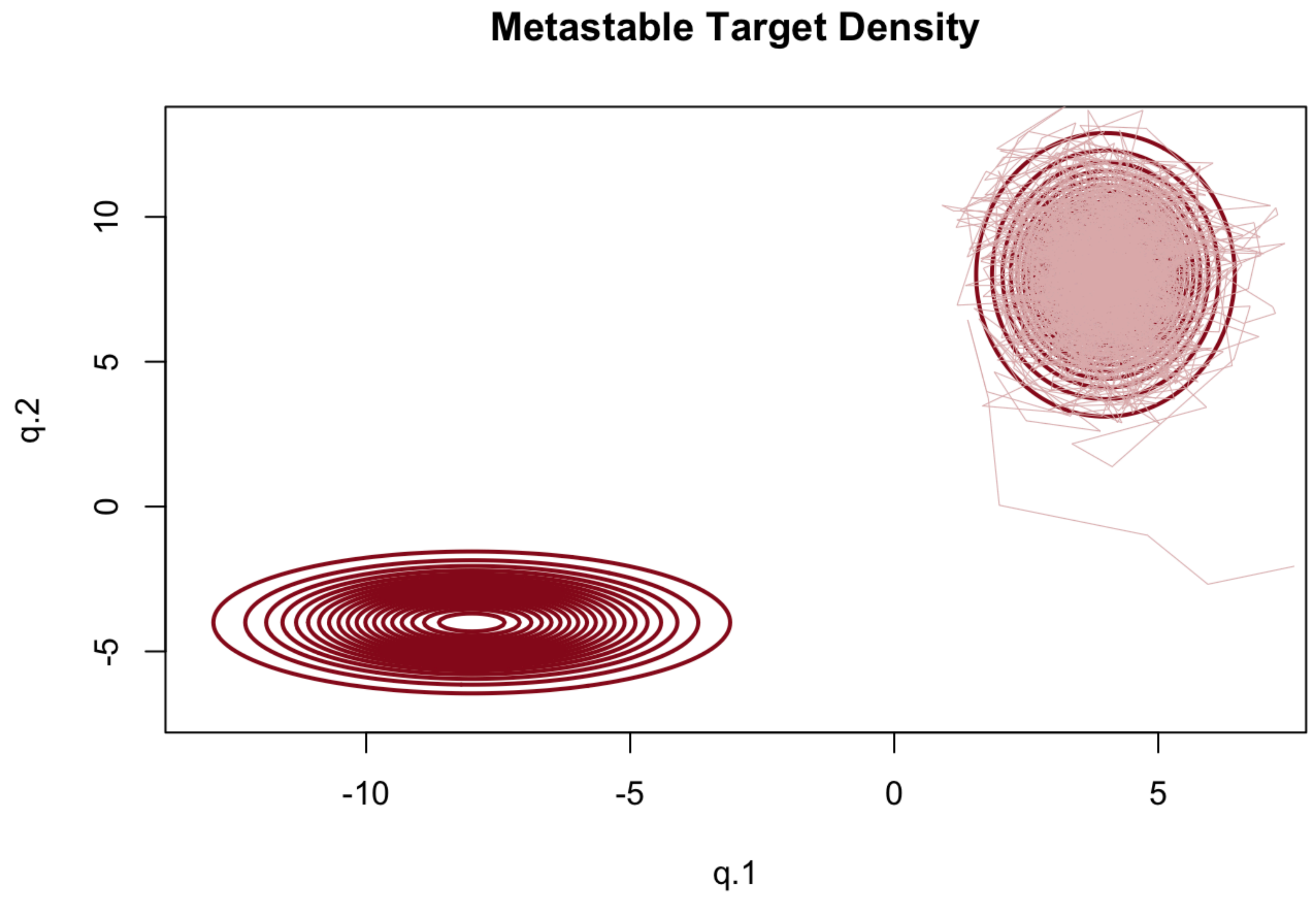
Sample correlation

- Next sample depends on the previous one
- Diagnostic tools ESS (effective sample size)
- Solution: thinning, ...

Pathological models

- Multimodal distribution
- Neal's Funnel





https://betanalpha.github.io/assets/case_studies/markov_chain_monte_carlo.html

Advanced Inference

Probabilistic Programming Languages

Hamiltonian Monte-Carlo (HMC)

Hamiltonian Monte-Carlo (HMC)

Analogy: Particle in an energy field

- Program define a density of the form $p(X) \propto \exp(-U(X))$
- On continuous spaces U can be interpreted as an energy
- Low energy wells correspond to high probability regions
- HMC simulate the trajectory of a particle in this energy field

Hamiltonian Monte-Carlo (HMC)

Analogy: Particle in an energy field

- Program define a density of the form $p(X) \propto \exp(-U(X))$
- On continuous spaces U can be interpreted as an energy
- Low energy wells correspond to high probability regions
- HMC simulate the trajectory of a particle in this energy field

Hamiltonian dynamic

- M: Mass matrix
- P: Momentum
- $K(P) = \frac{1}{2} P^T M P$

Hamiltonian Monte-Carlo (HMC)

Analogy: Particle in an energy field

- Program define a density of the form $p(X) \propto \exp(-U(X))$
- On continuous spaces U can be interpreted as an energy
- Low energy wells correspond to high probability regions
- HMC simulate the trajectory of a particle in this energy field

Hamiltonian dynamic

- M: Mass matrix
- P: Momentum
- $K(P) = \frac{1}{2} P^T M P$

$$\overset{\text{hamiltonian}}{\uparrow \atop H(X,P)} = \underset{\text{kinetic energy}}{K(P) \atop \downarrow} + \overset{\text{potential energy}}{\uparrow \atop U(X)}$$

Hamiltonian Monte-Carlo (HMC)

$$\underbrace{H(X,P)}_{\text{hamiltonian}} = \underbrace{K(P)}_{\text{kinetic energy}} + \underbrace{U(X)}_{\text{potential energy}}$$

Hamiltonian Monte-Carlo (HMC)

Energy conservation

$$\frac{dH}{dt} = (\nabla_P H)^T \frac{dP}{dt} + (\nabla_X H)^T \frac{dX}{dt} = 0$$

hamiltonian

potential energy

$$H(X, P) = K(P) + U(X)$$

kinetic energy

Hamiltonian Monte-Carlo (HMC)

Energy conservation

$$\frac{dH}{dt} = (\nabla_P H)^T \frac{dP}{dt} + (\nabla_X H)^T \frac{dX}{dt} = 0$$

Hamiltonian dynamics

$$\begin{cases} \dfrac{dX}{dt} = \nabla_P H(X, P) = M^{-1} P \\ \dfrac{dP}{dt} = \nabla_P H(X, P) = -\nabla_X U(X) \end{cases}$$

hamiltonian

potential energy

$$H(X, P) = K(P) + U(X)$$

kinetic energy

Hamiltonian Monte-Carlo (HMC)

Hamiltonian Monte-Carlo (HMC)

Generate samples from $p(X, P) \propto \exp(-H(X, P))$

- At each iteration
- Sample an initial momentum $P_i(0) \sim \mathcal{N}(0,\ M)$
- Solve the Hamiltonian dynamic (discretized)
- Perform a Metropolis Hastings update with probability α

$$\alpha = \min\left(1,\ \frac{\exp(-H(X_i, P_i))}{\exp(-H(X_{i-1}, P_{i-1}))}\right)$$

Hamiltonian Monte-Carlo (HMC)

Generate samples from $p(X, P) \propto \exp(-H(X, P))$

- At each iteration
- Sample an initial momentum $P_i(0) \sim \mathcal{N}(0,\ M)$
- Solve the Hamiltonian dynamic (discretized)
- Perform a Metropolis Hastings update with probability α

$$\alpha = \min\left(1,\ \frac{\exp(-H(X_i, P_i))}{\exp(-H(X_{i-1}, P_{i-1}))}\right)$$

If the hamiltonian is preserved: accept with probability 1.

Hamiltonian Monte-Carlo (HMC)

Generate samples from $p(X, P) \propto \exp(-H(X, P))$

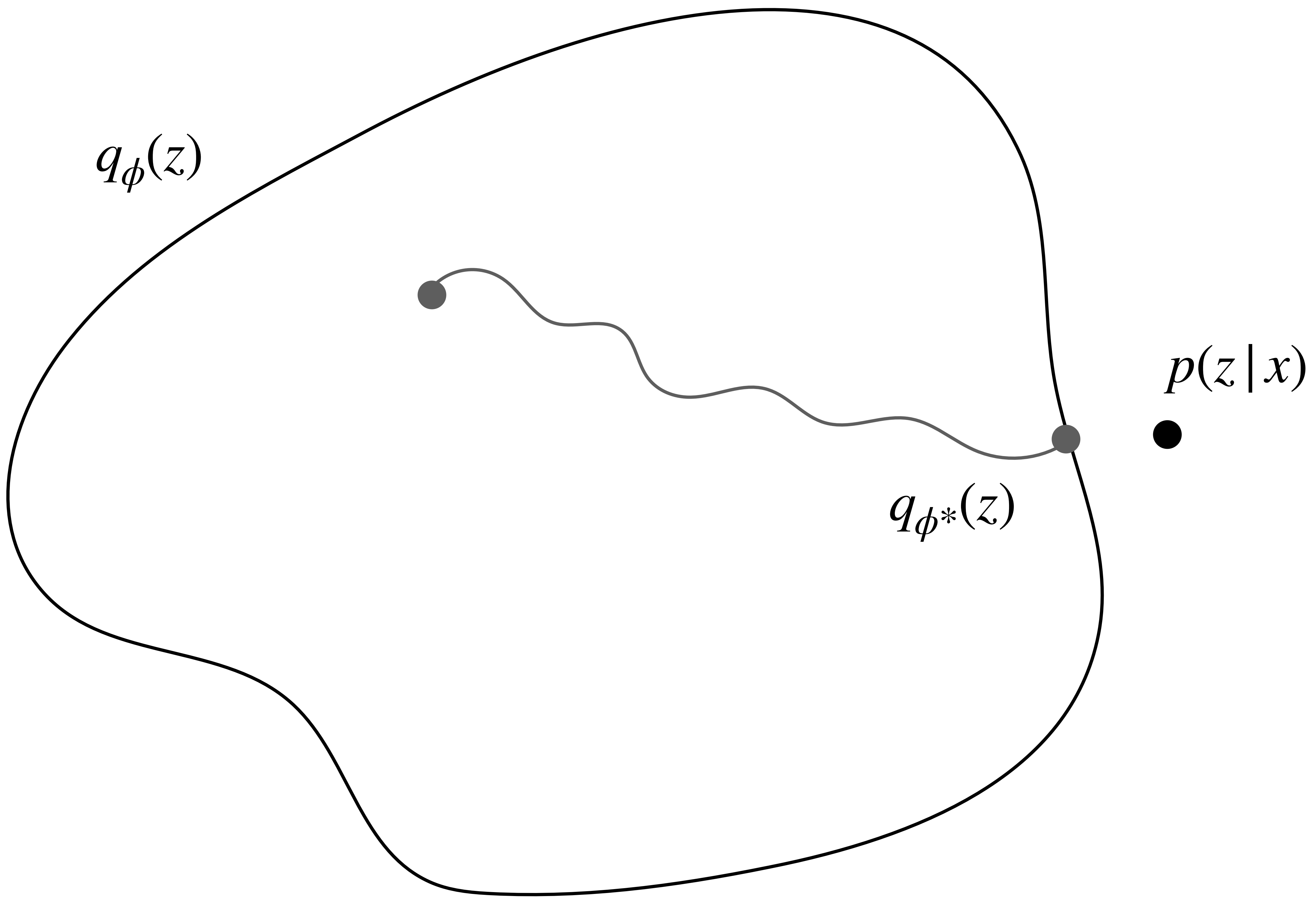
- At each iteration
- Sample an initial momentum $P_i(0) \sim \mathcal{N}(0,\ M)$
- Solve the Hamiltonian dynamic (discretized)
- Perform a Metropolis Hastings update with probability α

$$\alpha = \min\left(1,\ \frac{\exp(-H(X_i, P_i))}{\exp(-H(X_{i-1}, P_{i-1}))}\right)$$

If the hamiltonian is preserved: accept with probability 1.

Parameter P can be marginalized

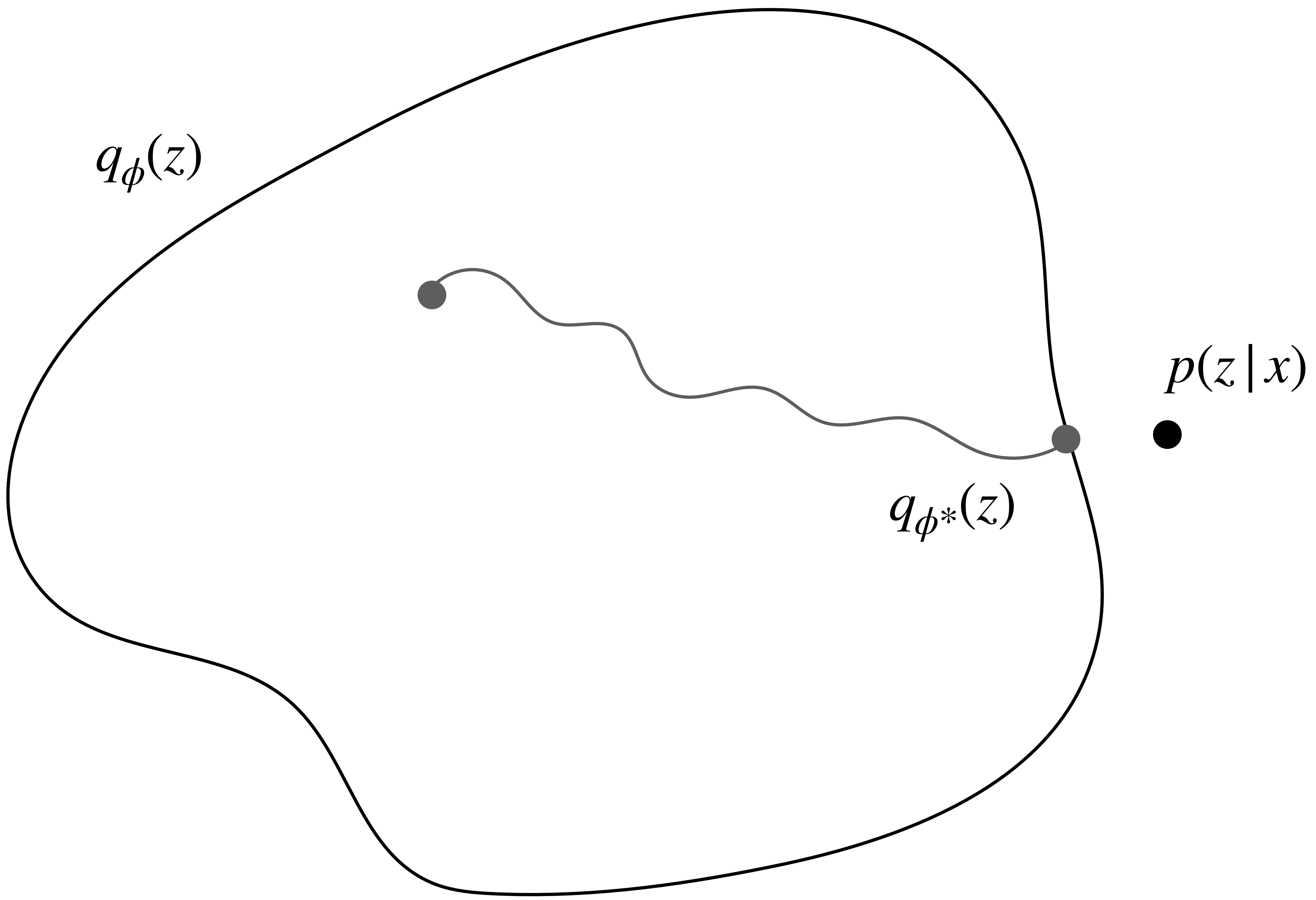
Stochastic Variational Inference (SVI)



Blei et al. 2017

Stochastic Variational Inference (SVI)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int_z p(x|z)p(z)dz}$$



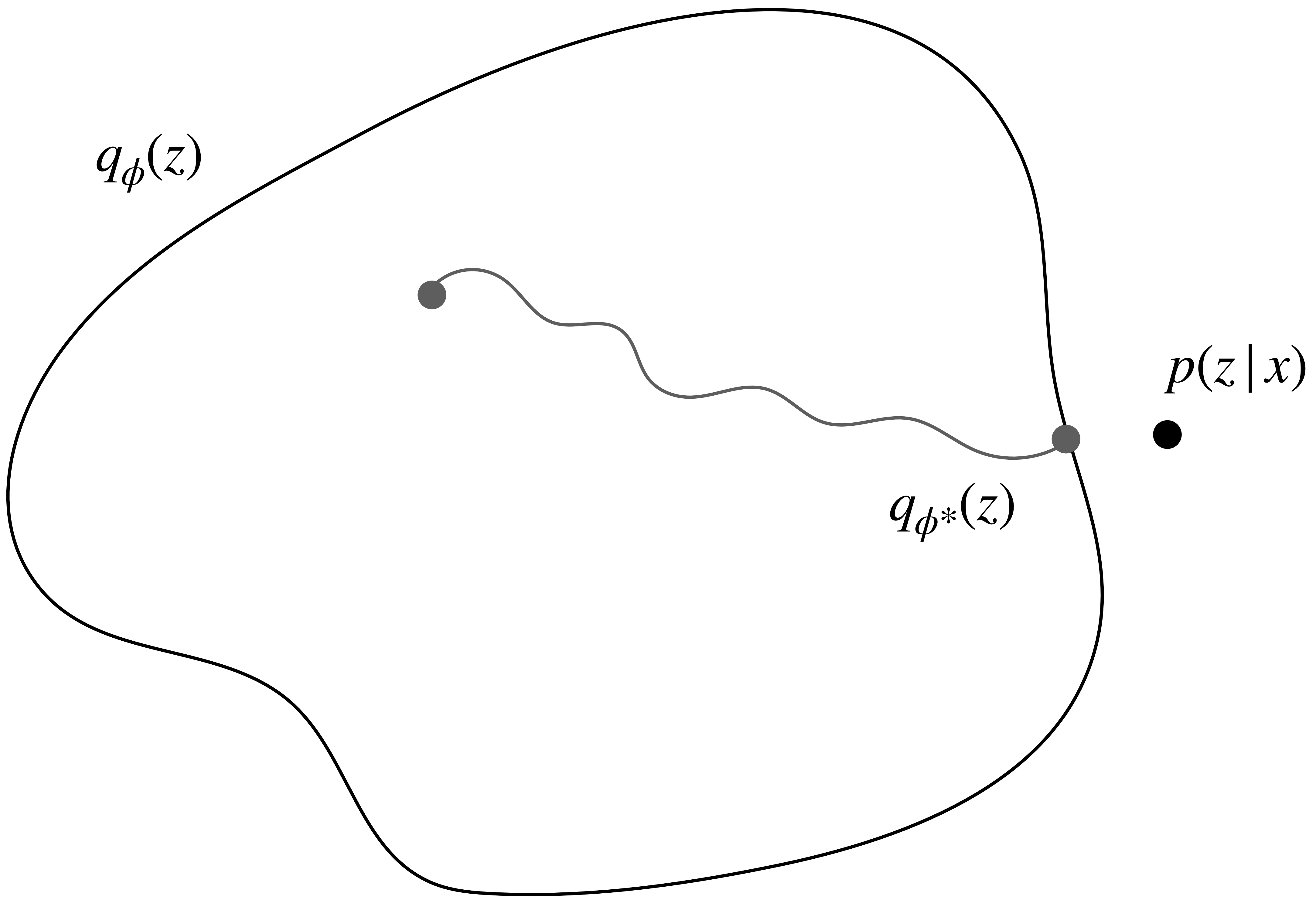
Blei et al. 2017

Stochastic Variational Inference (SVI)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int_z p(x|z)p(z)dz}$$

Variational family

- Parameterized by a parameter ϕ
- Find the closest member to the posterior $q_{\phi^*}(z)$
- Optimization problem



Blei et al. 2017

Stochastic Variational Inference (SVI)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int_z p(x|z)p(z)dz}$$

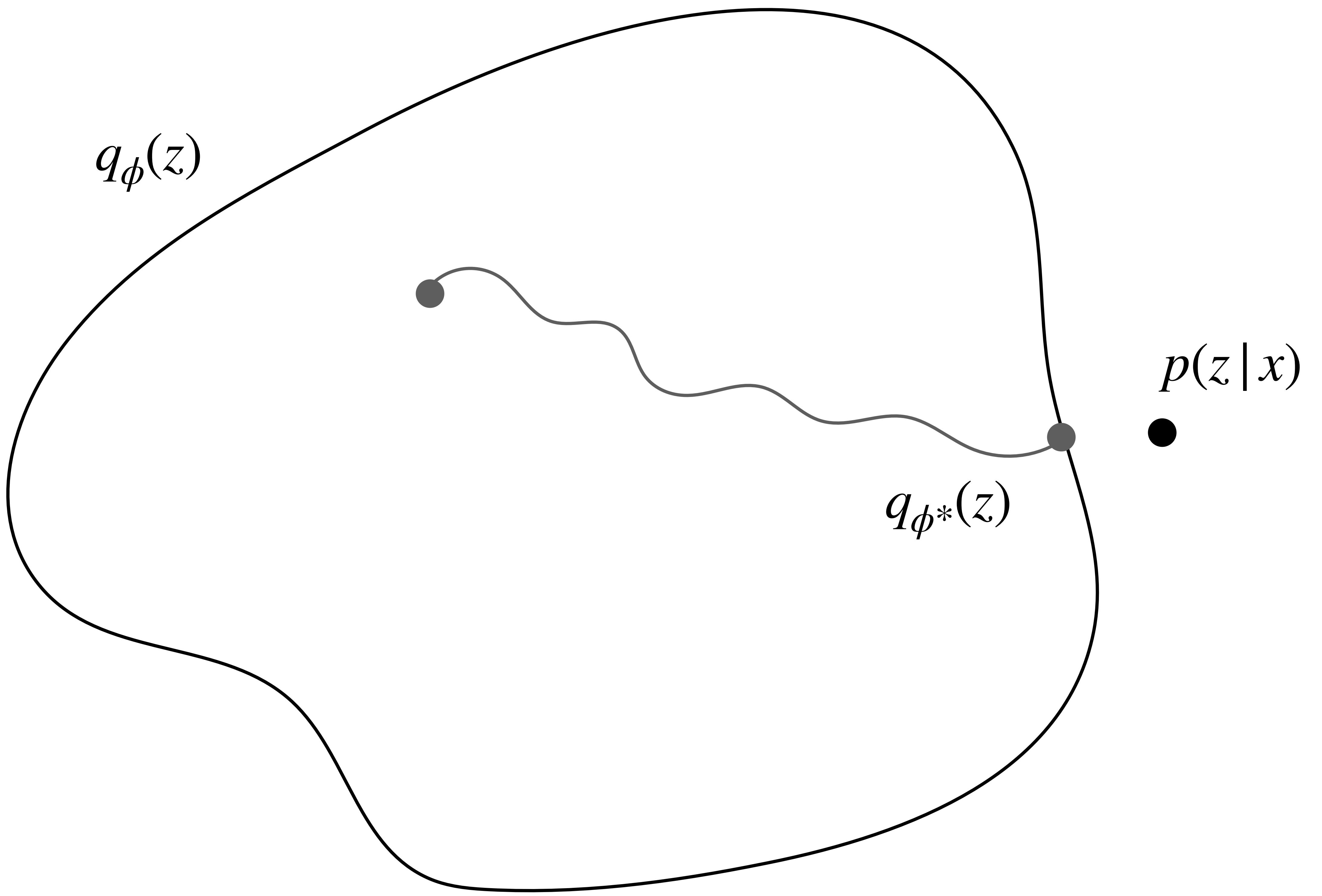
Variational family

- Parameterized by a parameter ϕ
- Find the closest member to the posterior $q_{\phi^*}(z)$
- Optimization problem

Metrics: Kullback-Leibler divergence

$$KL(q(x) \ || \ p(x)) = -\int q(x) \ \log \frac{p(x)}{q(x)} \ dx$$

- $KL(q \ || \ p) \geq 0$
- $KL(q \ || \ p) \neq KL(p \ || \ q)$



Blei et al. 2017

Stochastic Variational Inference (SVI)

$$
\begin{aligned}
KL(q_\phi(z) \mid\mid p(z|x)) &= -\int q_\phi(z) \log \frac{p(z|x)}{q_\phi(z)} \ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{p(x) q_\phi(z)} \ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \ dz + \int q_\phi(z) \log p(x) \ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \ dz + \log p(x)
\end{aligned}
$$

Stochastic Variational Inference (SVI)

$$
\begin{aligned}
KL(q_\phi(z)\ ||\ p(z|x)) &= -\int q_\phi(z) \log \frac{p(z|x)}{q_\phi(z)}\ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{p(x)q_\phi(z)}\ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)}\ dz + \int q_\phi(z) \log p(x)\ dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)}\ dz + \log p(x)
\end{aligned}
$$

$$
\log p(x) = KL(q_\phi(z)\ ||\ p(x,z)) + \int q_\phi(z)\ \log \frac{p(x,z)}{q_\phi(z)}\ dz
$$

Stochastic Variational Inference (SVI)

$$
\begin{aligned}
KL(q_\phi(z) \,||\, p(z|x)) &= -\int q_\phi(z) \log \frac{p(z|x)}{q_\phi(z)} \, dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{p(x) q_\phi(z)} \, dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \, dz + \int q_\phi(z) \log p(x) \, dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \, dz + \log p(x)
\end{aligned}
$$

$$
\underbrace{\log p(x)}_{\text{constant}} = KL(q_\phi(z) \,||\, p(x,z)) + \int q_\phi(z) \; \log \frac{p(x,z)}{q_\phi(z)} \, dz
$$

Stochastic Variational Inference (SVI)

$$
\begin{aligned}
KL(q_\phi(z) \,||\, p(z|x)) &= -\int q_\phi(z) \log \frac{p(z|x)}{q_\phi(z)}\; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{p(x)q_\phi(z)}\; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)}\; dz + \int q_\phi(z) \log p(x)\; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)}\; dz + \log p(x)
\end{aligned}
$$

$$
\underbrace{\log p(x)}_{\text{constant}} = \underbrace{KL(q_\phi(z) \,||\, p(x,z))}_{\text{minimize}} + \int q_\phi(z)\; \log \frac{p(x,z)}{q_\phi(z)}\; dz
$$

Stochastic Variational Inference (SVI)

$$
\begin{aligned}
KL(q_\phi(z) \mid\mid p(z|x)) &= -\int q_\phi(z) \log \frac{p(z|x)}{q_\phi(z)} \; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{p(x) q_\phi(z)} \; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \; dz + \int q_\phi(z) \log p(x) \; dz \\
&= -\int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} \; dz + \log p(x)
\end{aligned}
$$

$$
\underbrace{\log p(x)}_{\text{constant}} = \underbrace{KL(q_\phi(z) \mid\mid p(x,z))}_{\text{minimize}} + \underbrace{\int q_\phi(z) \; \log \frac{p(x,z)}{q_\phi(z)} \; dz}_{\text{maximize ELBO } \mathscr{L}}
$$

Stochastic Variational Inference (SVI)

$$\mathscr{L} = \int q_\phi(z) \ \log \frac{p(x,z)}{q_\phi(z)} \ dz$$

$$= \int q_\phi(z) \ \log \frac{p(x\,|\,z)p(z)}{q_\phi(z)} \ dz$$

$$= \int q_\phi(z) \ \log p(x\,|\,z) \ dz + \int q_\phi(z) \log \frac{p(z)}{q_\phi(z)} \ dz$$

$$\underbrace{\log p(x)}_{\text{constant}} = \underbrace{KL(q_\phi(z) \ || \ p(x,z))}_{\text{minimize}} + \underbrace{\int q_\phi(z) \ \log \frac{p(x,z)}{q_\phi(z)} \ dz}_{\text{maximize ELBO } \mathscr{L}}$$

Stochastic Variational Inference (SVI)

$$\mathscr{L} = \int q_\phi(z) \ \log \frac{p(x,z)}{q_\phi(z)} \ dz$$

$$= \int q_\phi(z) \ \log \frac{p(x|z)p(z)}{q_\phi(z)} \ dz$$

$$= \underbrace{\int q_\phi(z) \ \log p(x|z) \ dz}_{\substack{\text{maximize likelihood} \\ \mathbb{E}_q \log p(x|z)}} + \int q_\phi(z) \log \frac{p(z)}{q_\phi(z)} \ dz$$

$$\underbrace{\log p(x)}_{\text{constant}} = \underbrace{KL(q_\phi(z) \ || \ p(x,z))}_{\text{minimize}} + \underbrace{\int q_\phi(z) \ \log \frac{p(x,z)}{q_\phi(z)} \ dz}_{\text{maximize ELBO } \mathscr{L}}$$

Stochastic Variational Inference (SVI)

$$\mathscr{L} = \int q_\phi(z)\ \log \frac{p(x,z)}{q_\phi(z)}\ dz$$

$$= \int q_\phi(z)\ \log \frac{p(x|z)p(z)}{q_\phi(z)}\ dz$$

$$= \underbrace{\int q_\phi(z)\ \log p(x|z)\ dz}_{\substack{\text{maximize likelihood} \\ \mathbb{E}_q \log p(x|z)}} + \underbrace{\int q_\phi(z) \log \frac{p(z)}{q_\phi(z)}\ dz}_{\substack{\text{minimize divergence} \\ -KL(q(z)\ ||\ p(z))}}$$

$$\underbrace{\log p(x)}_{\text{constant}} = \underbrace{KL(q_\phi(z)\ ||\ p(x,z))}_{\text{minimize}} + \underbrace{\int q_\phi(z)\ \log \frac{p(x,z)}{q_\phi(z)}\ dz}_{\text{maximize ELBO } \mathscr{L}}$$

Variational Family

Variational Family

Black-box variational inference

- Mean-field approximation $q_\phi(z) = \prod_{i=1}^{n} \mathcal{N}(z_i \mid \mu_i, \sigma_i)$ where $\phi = \{\mu_i, \sigma_i\}_{i \in [1,n]}$
- Full-rank approximation $q_\phi(z) = \mathcal{N}(z \mid \mu, \Sigma)$ where $\phi = (\mu, \Sigma)$
- Pyro autoguides

Variational Family

Black-box variational inference

- Mean-field approximation $q_\phi(z) = \prod_{i=1}^{n} \mathcal{N}(z_i \mid \mu_i, \sigma_i)$ where $\phi = \{\mu_i, \sigma_i\}_{i \in [1,n]}$
- Full-rank approximation $q_\phi(z) = \mathcal{N}(z \mid \mu, \Sigma)$ where $\phi = (\mu, \Sigma)$
- Pyro autoguides

Program your own guide

- Pyro (first versions)
- Must sample the same variables in the guide and the model
- Static analysis?

```
def model():
    pyro.sample("z_1", ...)

def guide():
    pyro.sample("z_1", ...)
```


Lee, Yu, Rival, Yang POPL 2020

References

An Introduction to Probabilistic Programming
Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, Frank Wood
https://arxiv.org/abs/1809.10756

Lightweight Implementations of Probabilistic Programming Languages Via Transformational Compilation
David Wingate Andreas Stuhlmüller Noah D. Goodman
AISTATS 2011

Markov Chain Monte Carlo in Practice
Michael Bettancourt
https://betanalpha.github.io/assets/case_studies/markov_chain_monte_carlo.html

Variational Inference: A Review for Statisticians
David M. Blei, Alp Kucukelbir, Jon D. McAuliffe
https://arxiv.org/abs/1601.00670

Compiling Stan to Generative Probabilistic Languages and Extension to Deep Probabilistic Programming
Guillaume Baudart, Javier Burroni, Martin Hirzel, Louis Mandel, Avraham Shinnar
PLDI 2021