

CS492: Probabilistic Programming

Markov Chain

Monte Carlo

Hongseok Yang

KAIST

CS492: Probabilistic Programming

Markov Chain

Monte Carlo



Hongseok Yang  
KAIST

Really about: Metropolis-Hastings algorithm

```
(doquery :lmh induce-fn [ints2 outs2])
```

(doquery :lmh induce-fn [ints2 outs2])



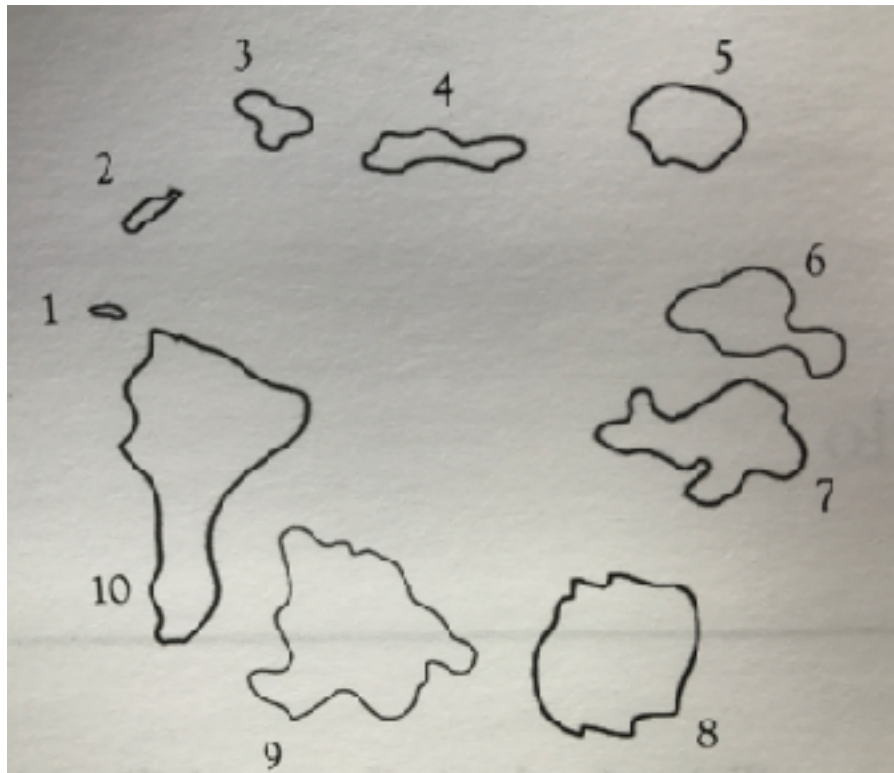
Lightweight Metropolis Hastings algorithm\* (LMH).

\* Wingate, Stuhlmuller and Goodman's paper at AISTATS 2011

# Learning outcome

- Can explain Metropolis-Hastings algorithm.
- Can say when this algo. is correct.
- Can develop an instance of the algorithm.

# Good king Markov puzzle\*

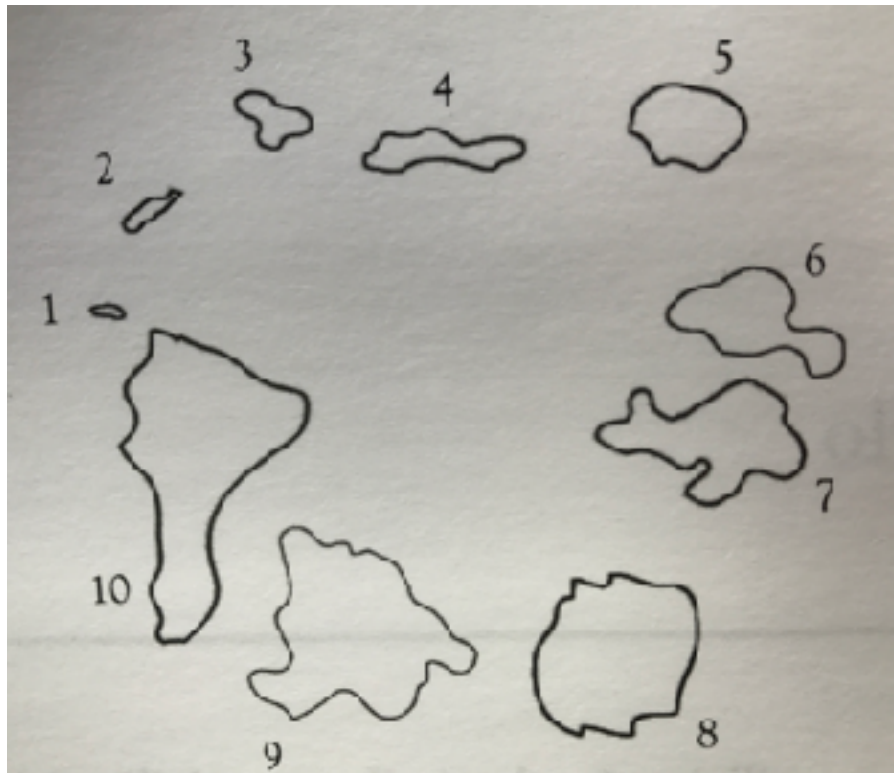


Markov rules 10 islands.

$100i$  people live in island  $i$ .

\* Borrowed from McElreath's book "Statistical Rethinking"

# Good king Markov puzzle\*



Markov rules 10 islands.

$100i$  people live in island  $i$ .

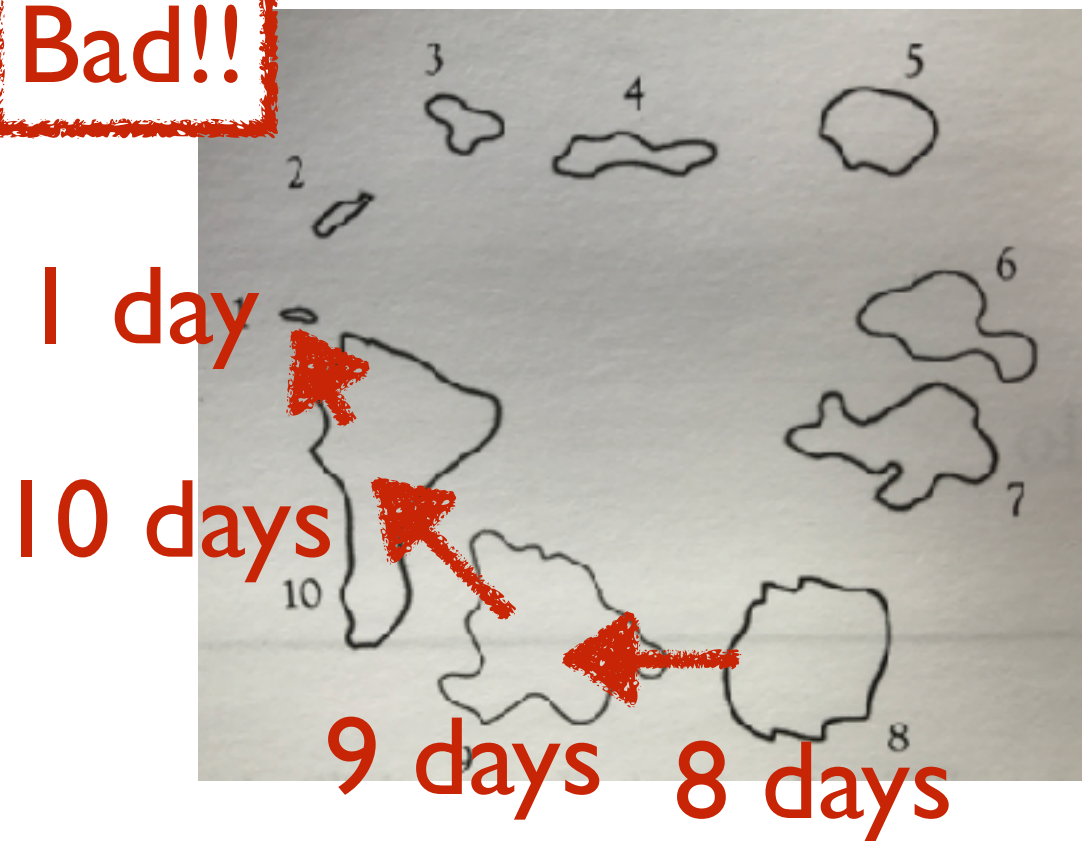
King loves his people and wants to visit each island in proportion to its population size.

[Q] Find an algorithm. No scheduling nor bookkeeping. Can move adjacent islands only.

\* Borrowed from McElreath's book "Statistical Rethinking"

# Good king Markov puzzle\*

Bad!!



Markov rules 10 islands.

$100i$  people live in island  $i$ .

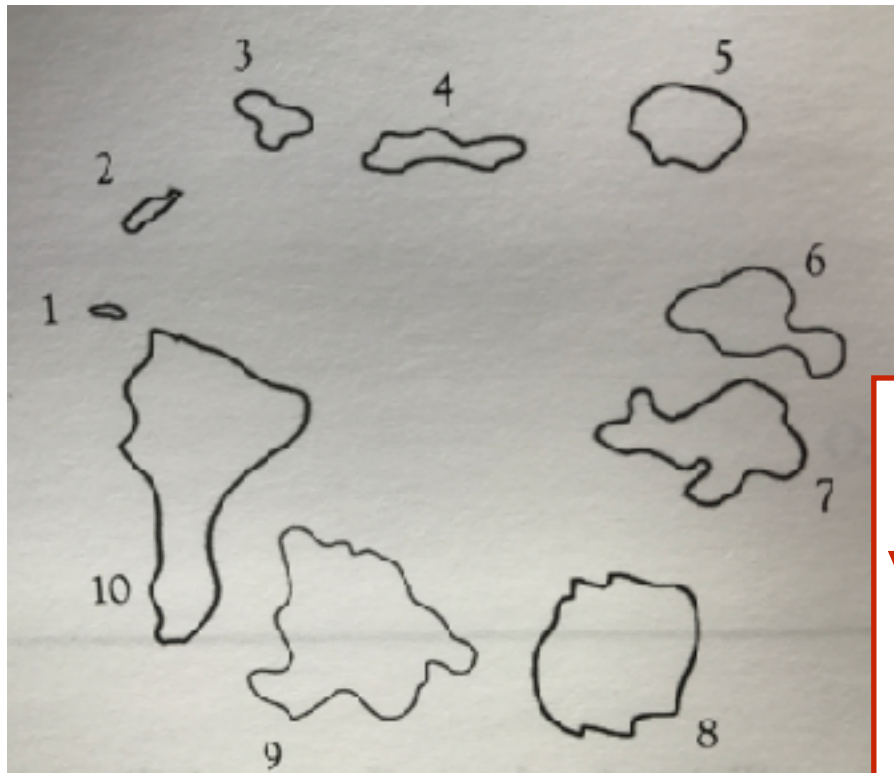
King loves his people and wants to visit each island in proportion to its population size.

[Q] Find an algorithm. **No scheduling nor bookkeeping.** Can move adjacent islands only.

\* Borrowed from McElreath's book "Statistical Rethinking"



# Good king Markov puzzle\*



Markov rules 10 islands.

$100i$  people live in island  $i$ .

$i \sim \text{discrete}(1, 2, \dots, 10)$ .  
Visit  $i$ .  
Repeat.

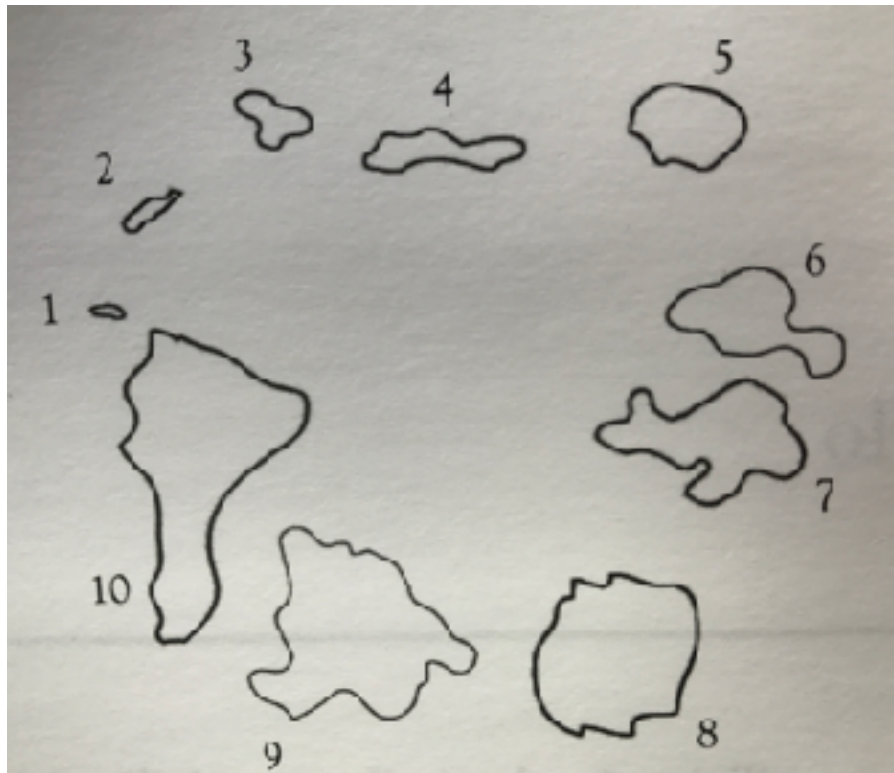
Bad!!

King loves his people and wants to visit each island in proportion to its population size.

[Q] Find an algorithm. No scheduling nor bookkeeping. **Can move adjacent islands only.**

\* Borrowed from McElreath's book "Statistical Rethinking"

# Good king Markov puzzle\*



Markov rules 10 islands.

$100i$  people live in island  $i$ .

King loves his people and wants to visit each island in proportion to its population size.

**[Q] Find an algorithm. No scheduling nor bookkeeping. Can move adjacent islands only.**

\* Borrowed from McElreath's book "Statistical Rethinking"

# Solution

$k_n$  — island that the king visits at step  $n$ .

Repeat the following steps:

1. Flip a coin with prob. 0.5. If head, pick next  $k'$  clockwise. If tail, use  $k'$  counterclockwise.
2.  $\alpha := \min(1, k'/k_n)$ .
3. Flip a coin with prob.  $\alpha$ . If head,  $k_{n+1} := k'$ . Otherwise,  $k_{n+1} := k_n$ .

# Solution

$k_n$  — island that the king visits at step  $n$ .

Repeat the following steps:

1. Flip a coin with prob. 0.5. If head, pick next  $k'$  clockwise. If tail, use  $k'$  counterclockwise.
2.  $\alpha := \min(1, k'/k_n)$ .
3. Flip a coin with prob.  $\alpha$ . If head,  $k_{n+1} := k'$ . Otherwise,  $k_{n+1} := k_n$ .

[Q] Why correct? What does correctness even mean?

Sequence by the algo.:  $k_1, k_2, \dots, k_n, \dots$

Sequence by the algo.:  $k_1, k_2, \dots, k_n, \dots$

[Strong convergence] For any  $f : \{1, \dots, 10\} \rightarrow \mathbb{R}$ ,  
 $(\sum_{j \leq n} f(k_j))/n \longrightarrow \mathbb{E}_{p(i)}[f(i)]$  as  $n \longrightarrow \infty$  with prob. 1,  
where  $p(i) = i/55$ , target prob. for visiting island  $i$ .

Intuitively, frequency represents probability.

Sequence by the algo.:  $k_1, k_2, \dots, k_n, \dots$

[Strong convergence] For any  $f : \{1, \dots, 10\} \rightarrow \mathbb{R}$ ,  
 $(\sum_{j \leq n} f(k_j))/n \longrightarrow \mathbb{E}_{p(i)}[f(i)]$  as  $n \longrightarrow \infty$  with prob. 1,  
where  $p(i)=i/55$ , target prob. for visiting island  $i$ .

Intuitively, frequency represents probability.

Holds because 1) the random move of the algo.  
has  $p$  as **invariant**; 2) the algo. can **move between  
any two islands** in finitely many steps.

Sequence by the algo.:  $k_1, k_2, \dots, k_n, \dots$

[Strong convergence] For any  $f : \{1, \dots, 10\} \rightarrow \mathbb{R}$ ,  
 $(\sum_{j \leq n} f(k_j))/n \longrightarrow \mathbb{E}_{p(i)}[f(i)]$  as  $n \longrightarrow \infty$  with prob. 1,  
where  $p(i) = i/55$ , target prob. for visiting island  $i$ .

Intuitively, frequency represents probability.

Holds because 1) the random move of the algo.  
has  $p$  as invariant; 2) the algo. can move between  
any two islands in finitely many steps.

[Q] Prove 1) and 2).



# Metropolis algorithm

Goal: Generate samples from target  $r(x)/Z$ , where  $Z = \int r(x)dx$ , the normalising constant.

Parameter: Conditional distribution  $q(x'|x)$ .

- Should be symmetric  $q(x'|x) = q(x|x')$ .
- Represents a random move.
- Called proposal kernel.

# Metropolis algorithm

Goal: Generate samples from target  $r(x)/Z$ , where  $Z = \int r(x)dx$ , the normalising constant.

Parameter: Conditional distribution  $q(x'|x)$ .

- Should be symmetric  $q(x'|x) = q(x|x')$ .
- Represents a random move.
- Called proposal kernel.

[Q] What are  $r(-)$  and  $q(-|-)$  in King Markov?

# Metropolis algorithm

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

# Metropolis

Noisy greedy exploration.

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$   
 $\geq 1$  for better  $x'$   
 $< 1$  for worse  $x'$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

# Metropolis

Noisy greedy exploration.  
No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

# Metropolis

Noisy greedy exploration.  
No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q1] How is it related to our sol. for King Markov?

# Metropolis

Noisy greedy exploration.  
No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q2] Does each step preserve  $r(x)/Z$  as invariant?

# Metropolis

Noisy greedy exploration.  
No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q3] Use it & do posterior inference. Assume  $x \in \mathbb{R}^m$ .



$$r(x) = p(y|x)p(x)$$

$$q(x'|x) = \text{normal}(x, \varepsilon \times \text{ID})$$

Noisy greedy exploration.

No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q3] Use it & do posterior inference. Assume  $x \in \mathbb{R}^m$ .

# Metropolis

Noisy greedy exploration.  
No need to know  $Z$ .

Target  $r(x)/Z$ . Symmetric proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$
2. repeat:
  - a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, r(x')/r(x_n))$
  - b)  $u \sim \text{uniform}(0,1)$
  - c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q4] Instantiate this algo. for Anglican programs.

```
(let [x1 (sample (normal 0 1))  
      x2 (if (> x1 0)  
            1  
            (sample (normal (* x1 x1) 4)))]  
  (observe (normal x2 1) 3)  
  (list x1 x2))
```

```
(let [x1 (sample (normal 0 1))  
      x2 (if (> x1 0)  
            1  
            (sample (normal (* x1 x1) 4)))]  
  (observe (normal x2 1) 3)  
  (list x1 x2))
```

Option 1: Use density formula for all random vars.

```
(let [x1 (sample (normal 0 1))  
      x2 (if (> x1 0)  
            1  
            (sample (normal (* x1 x1) 4)))]  
  (observe (normal x2 1) 3)  
  (list x1 x2))
```

Option 1: Use density formula for all random vars.

```
(let [x1 (sample (normal 0 1))
      x2 (if (> x1 0)
            1
            (sample (normal (* x1 x1) 4)))]
  (observe (normal x2 1) 3)
  (list x1 x2))
```

Option 1: Use density formula for all random vars.

Can't  
handle  
loops.

```
(let [p (sample (uniform-continuous 0.1 0.9))
      n (loop [i 0]
            (if (sample (flip p)) i (recur (+ i 1))))]
  (observe (normal n 3) 5)
  p)
```

```
(let [x1 (sample (normal 0 1))  
      x2 (if (> x1 0)  
            1  
            (sample (normal (* x1 x1) 4)))]  
  (observe (normal x2 1) 3)  
  (list x1 x2))
```

**Option 1: Use density formula for all random vars.**

**Can't  
handle  
loops.**

```
(let [p (sample (uniform-continuous 0.1 0.9))  
      n (loop [i 0]  
            (if (sample (flip p)) i (recur (+ i 1))))]  
  (observe (normal n 3) 5)  
  p)
```

**Option 2: Use distribution on execution traces.**

```
(let [x1 (sample (normal 0 1))
      x2 (if (> x1 0)
            1
            (sample (normal (* x1 x1) 4)))]
  (observe (normal x2 1) 3)
  (list x1 x2))
```

Option 1: Use density formula for all random vars.

Can't  
handle  
loops.

```
(let [p (sample (uniform-continuous 0.1 0.9))
      n (loop [i 0]
            (if (sample (flip p)) i (recur (+ i 1))))]
  (observe (normal n 3) 5)
  p)
```

Option 2: Use distribution on execution traces.  
But difficult to find symmetric  $q$ .



# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n|x')}{r(x_n) \times q(x'|x_n)})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n); \quad n:=n+1$

# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n|x')}{r(x_n) \times q(x'|x_n)})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n); \quad n:=n+1$

[Q1] Develop  $q$  for Anglican programs.

# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n|x')}{r(x_n) \times q(x'|x_n)})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n); \quad n:=n+1$

[Q2] Noisy greedy exploration. Find a (relative) obj.

# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n|x')}{r(x_n) \times q(x'|x_n)})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n); \quad n:=n+1$

[Q2] Noisy greedy exploration. Find a (relative) obj.

# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ prop

When independent  
proposal is used.

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n)}{r(x_n) \times q(x')})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n); \quad n:=n+1$

[Q2] Noisy greedy exploration. Find a (relative) obj.

# Hastings Metropolis algorithm

Target  $r(x)/Z$ . ~~Symmetric~~ proposal  $q(x'|x)$ .

1. initialise  $x_1$  randomly;  $n:=1$

2. repeat:

a)  $x' \sim q(x'|x_n)$ ;  $\alpha := \min(1, \frac{r(x') \times q(x_n|x')}{r(x_n) \times q(x'|x_n)})$

b)  $u \sim \text{uniform}(0,1)$

c)  $x_{n+1} := (\text{if } (u \leq \alpha) \text{ then } x' \text{ else } x_n)$ ;  $n:=n+1$

[Q3] Does each step preserve  $r(x)/Z$  as invariant?

# Recap of the MH algo.

- Generate samples from unnormalised  $r(x)$ .  
No need to know  $Z = \int r(x)dx$ .
- Noisy greedy exploration using  $q(x'|x)$ .

# Guarantees informally

[Thm I] Each step of MH has  $r/Z$  as inv. dist.



# Guarantees informally

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

[Thm2] For all  $f: X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

# Guarantees informally

The estimate converges to the right value.

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

[Thm2] For all  $f:X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

# Guarantees informally

The estimate converges to the right value.

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

[Thm2] For all  $f: X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

MH moves well. For any  $r/Z$ -possible  $x, x'$ , the MH can go from  $x$  to  $x'$  with non-zero prob.

# Guarantees informally

The estimate converges to the right value.

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

[Thm2] For all  $f: X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

MH moves well. For any  $r/Z$ -possible  $x, x'$ , the MH can go from  $x$  to  $x'$  with non-zero prob.

# Guarantees informally

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

[Thm2] For all  $f: X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

# Guarantees informally

MH samples:  $x_1, x_2, x_3, \dots, x_n, \dots$

**[Thm2]** For all  $f: X \rightarrow \mathbb{R}$  with  $\mathbb{E}_{r(x)/Z}[f(x)]$  defined,

$\sum_{i \leq n} f(x_i)/n \longrightarrow \mathbb{E}_{r(x)/Z}[f(x)]$  as  $n \longrightarrow \infty$  with prob. 1,

if the MH with  $q$  is  $r/Z$ -irreducible.

Consequence of a general result in ergodic theory.  
Thm1 plays a crucial role in the proof.

# Reference

I looked at Chapters 5 and 6 of Robert & Casella's "Monte Carlo Statistical Methods".

Not recommended for general reading.

But details and pointers can be found there.