

Statistical modeling with stochastic processes

Alexandre Bouchard-Côté
Lecture 4, Monday March 8

Program for today

- Wrapping up MCMC
 - Auxiliary variables and collapsing
 - Common errors
 - Annealing and tempering
 - Infinite spaces
- Approximate inference, Part 2: Variational
 - Examples: Mean field and Belief propagation
 - Theoretical framework

Review

Question

How to build T such that:

$$\text{statio}(x) = \text{target}(x)$$

First step: finding a better expression for $\text{statio}(x)$

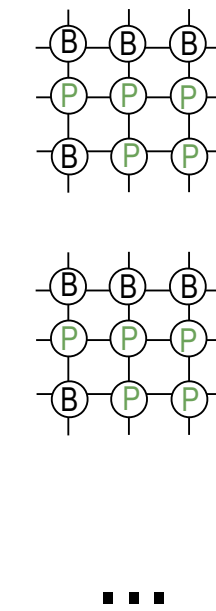
$$\text{statio}(x) = \lim_{t \rightarrow \infty} \mathbb{P}(X_t = x | X = 0 = *)$$

$$\text{target}(x) = \mathbb{P}(X = x | \text{obs, params})$$

Finding a better expression for $\text{statio}(x)$

Definition ('infinite steps' transition): $T^\infty = \lim_{n \rightarrow \infty} T^n$

Hope:

$$T^\infty = \left[\begin{array}{ccc} \text{---} & \pi & \text{---} \\ \text{---} & \pi & \text{---} \\ & \vdots & \\ \text{---} & \pi & \text{---} \end{array} \right]$$


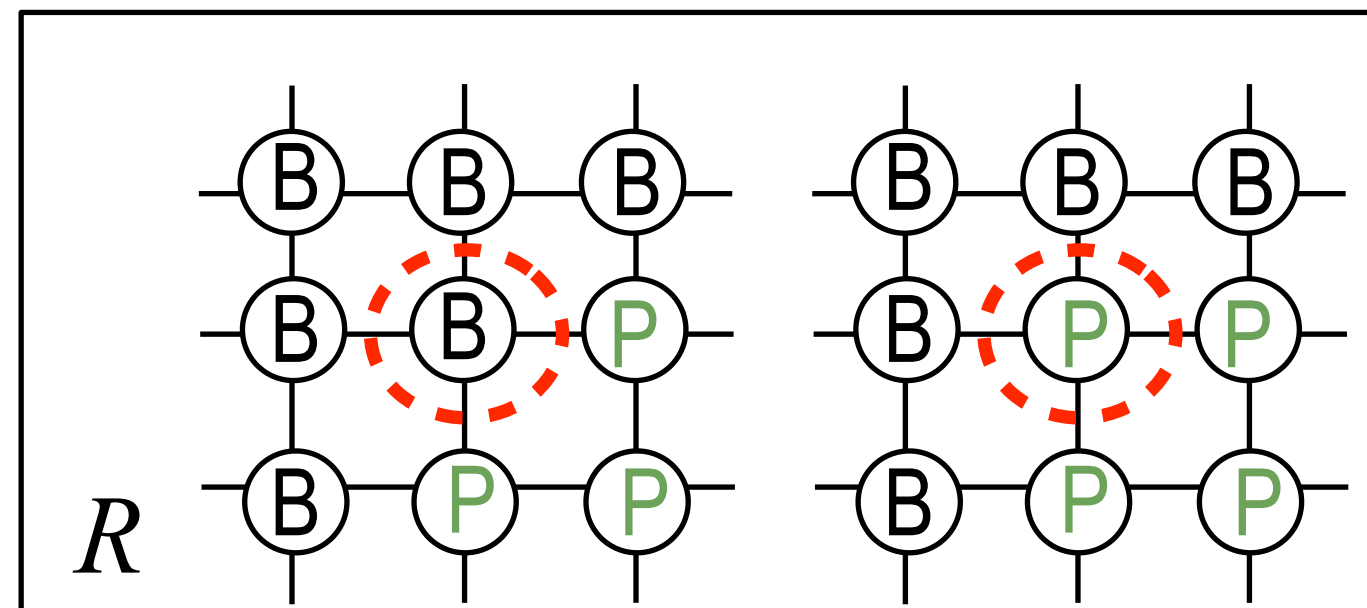
That would mean that no matter what state we use to initialize the sampler, the distribution over the n -th state converges to a distribution called the stationary distribution $\pi(x) = \text{statio}(x) = \text{target}(x)$

Building T such that $\text{statio}(x) = \text{target}(x)$

Goal: Let's see if Gibbs satisfies this equation

$$\text{target}(x) = \sum_y \text{target}(y) T_{y,x} \quad (1)$$

First: Let's find what is $T_{y,x}$



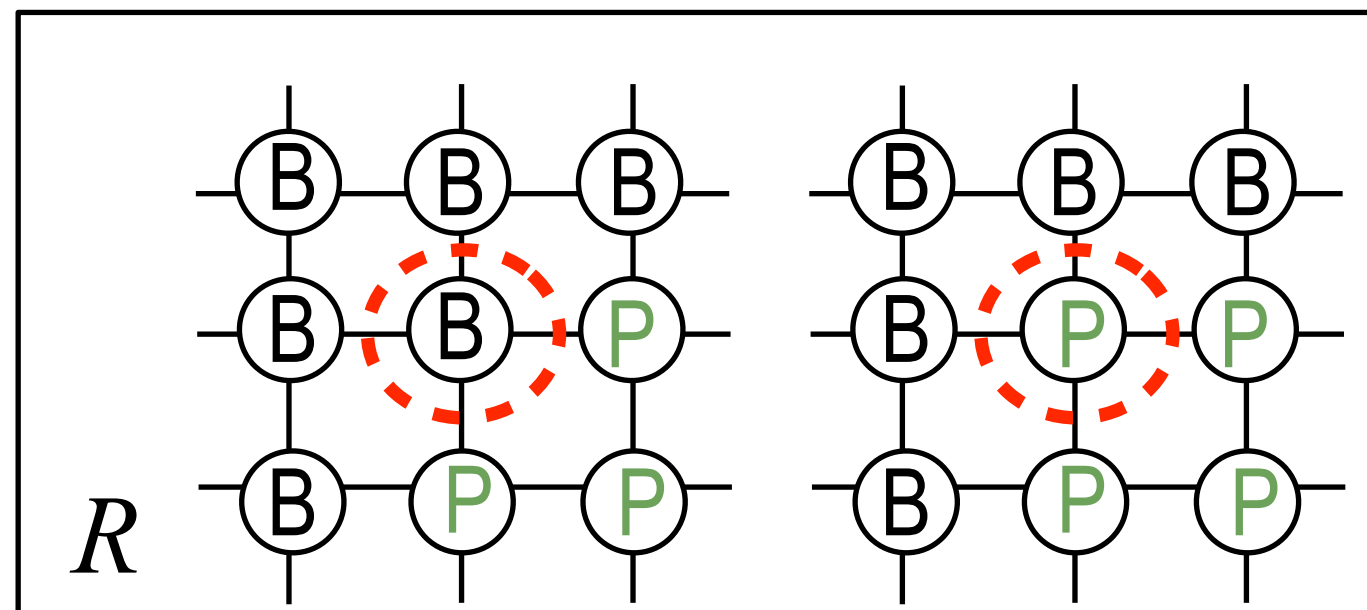
Building T such that $\text{statio}(x) = \text{target}(x)$

Goal: Let's see if Gibbs satisfies this equation

$$\text{target}(x) = \sum_y \text{target}(y) T_{y,x} \quad (1)$$

First: Let's find what is $T_{y,x}$

$$T_{y,x} = \frac{\mathbf{1}[x, y \in R] \text{target}(x)}{\sum_{x'} \mathbf{1}[x', y \in R] \text{target}(x')}$$



Building T such that $\text{statio}(x) = \text{target}(x)$

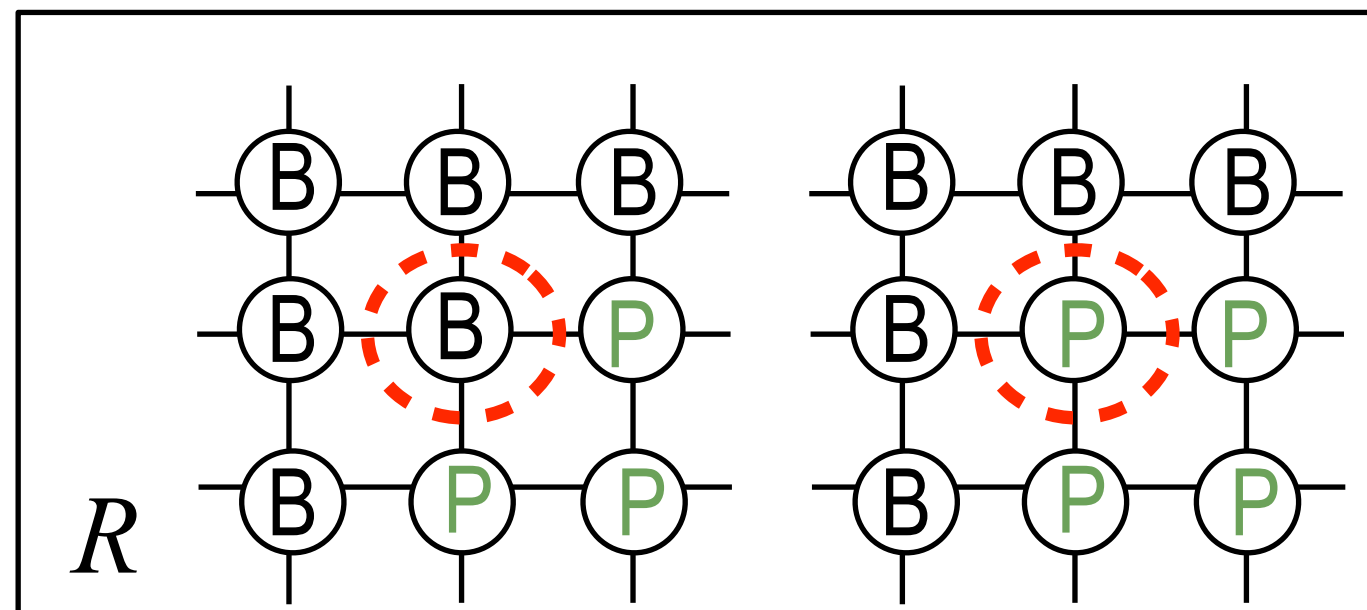
Goal: Let's see if Gibbs satisfies this equation

$$\text{target}(x) = \sum_y \text{target}(y) T_{y,x} \quad (1)$$

First: Let's find what is $T_{y,x}$

$$T_{y,x} = \frac{\mathbf{1}[x, y \in R] \text{target}(x)}{\sum_{x'} \mathbf{1}[x', y \in R] \text{target}(x')} \quad (2)$$

Finally: plug-in (2) in (1)
and check it works



Why Metropolis-Hastings works

From previous result, want T such that:

$$\text{target}(x) = \sum_y \text{target}(y) T_{y,x}$$

Sufficient condition (by summing over y on both sides):

$$\text{target}(x) T_{x,y} = \text{target}(y) T_{y,x}$$

This is called *detailed balance* or *reversibility* condition

Existence of π such that $\pi = \pi T$

Suppose: (still assuming discrete state space)

1. T is irreducible
2. T is aperiodic

Consequence: There is a unique probability distribution π such that $\pi = \pi T$

Proofs: Consequence of Perron–Frobenius theorem (T^n is positive for n large enough, and π is then the eigenvector corresponding to the unique eigenvalue of highest modulus). --- **Note:** can be used to debug samplers

More general arguments exist

Convergence theorem 1

Suppose: (still assuming discrete state space)

1. T is irreducible
2. T is aperiodic

Consequence: There is a unique probability distribution π such that $\pi = \pi T$; moreover, for all x ,

$$\lim_{n \rightarrow \infty} T_{x,y}^n = \pi(y)$$

i.e.:

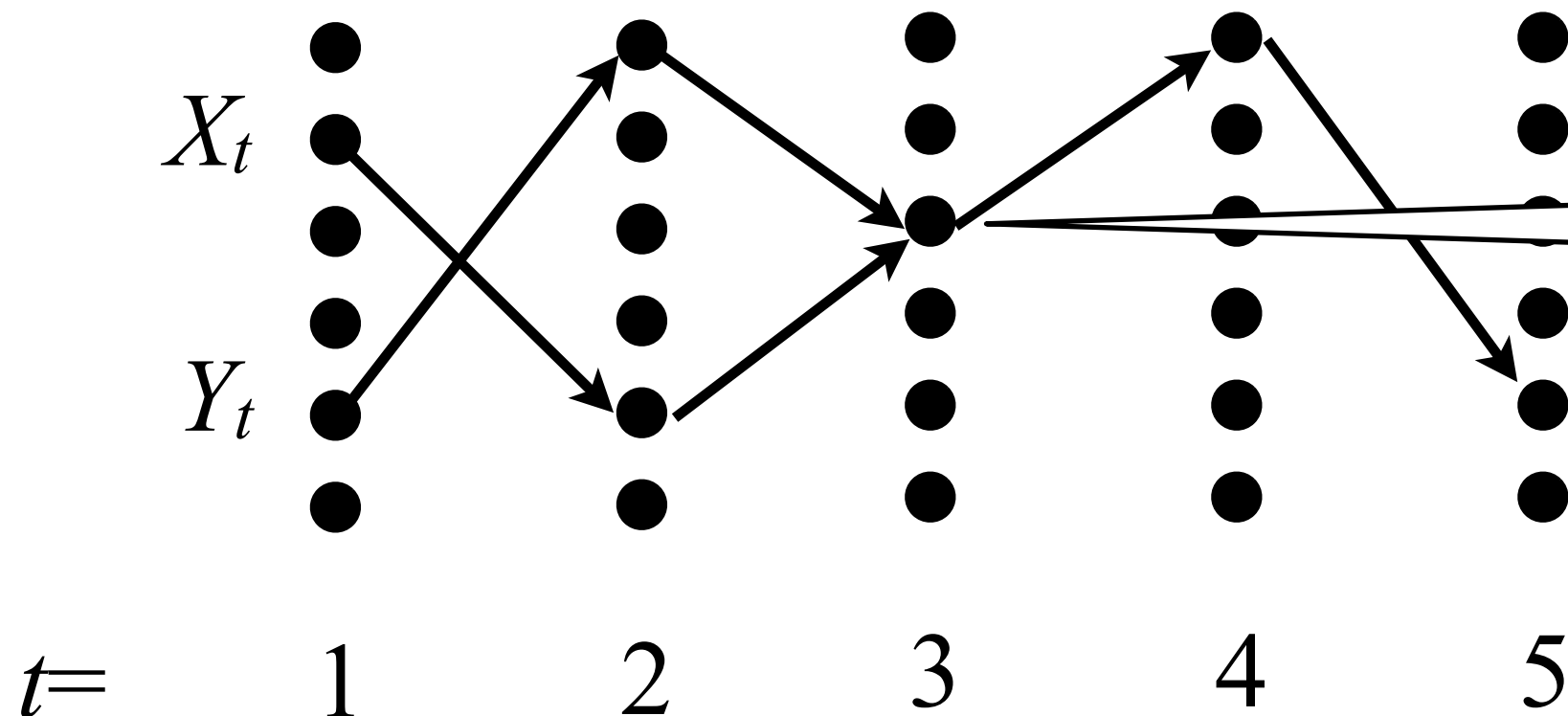
$$T^\infty = \begin{bmatrix} \text{---} \pi \text{---} \\ \text{---} \pi \text{---} \\ \vdots \\ \text{---} \pi \text{---} \end{bmatrix}$$

Proof: coupling argument

Initial distributions: $X_0 \sim \pi$ and $Y_0 \sim$ arbitrary distribution

Note: $X_t \sim \pi$ for all t since $\pi = \pi T$

Goal: showing that $\lim_{n \rightarrow \infty} \sum_y \left| \mathbb{P}(X_n = y) - \mathbb{P}(Y_n = y) \right| = 0$



Notation: the hitting time H where $X_t = Y_t$ for the first time.
e.g. $H=3$ here

LLN for Markov chains

The law of large numbers for Markov chains: If X_t is an irreducible Markov chain with stationary distribution π and f is finite, then

$$\lim_{n \rightarrow \infty} \frac{1}{S} \sum_{t=1}^S f(X_t) = \sum_x f(x) \pi(x)$$

Note 1: Aperiodicity not needed for this result

Note 2: For small S , burning-in might improve the estimator, but might as well maximize during burn-in

Note 2: Thinning to reduce auto-correlation is not a good idea and can be harmful (only reasons to do it is to save memory writes or memory---but most of the time only finite dimensional sufficient statistics need to be stored)

Wrapping-up MCMC

Terminology

Collapsed sampler: analytically marginalize some of the variables, and run MCMC on the *reduced* state space (makes sampling harder/more expensive, but improves the quality of the samples)

Auxiliary variable: *augment* the state space to facilitate sampling

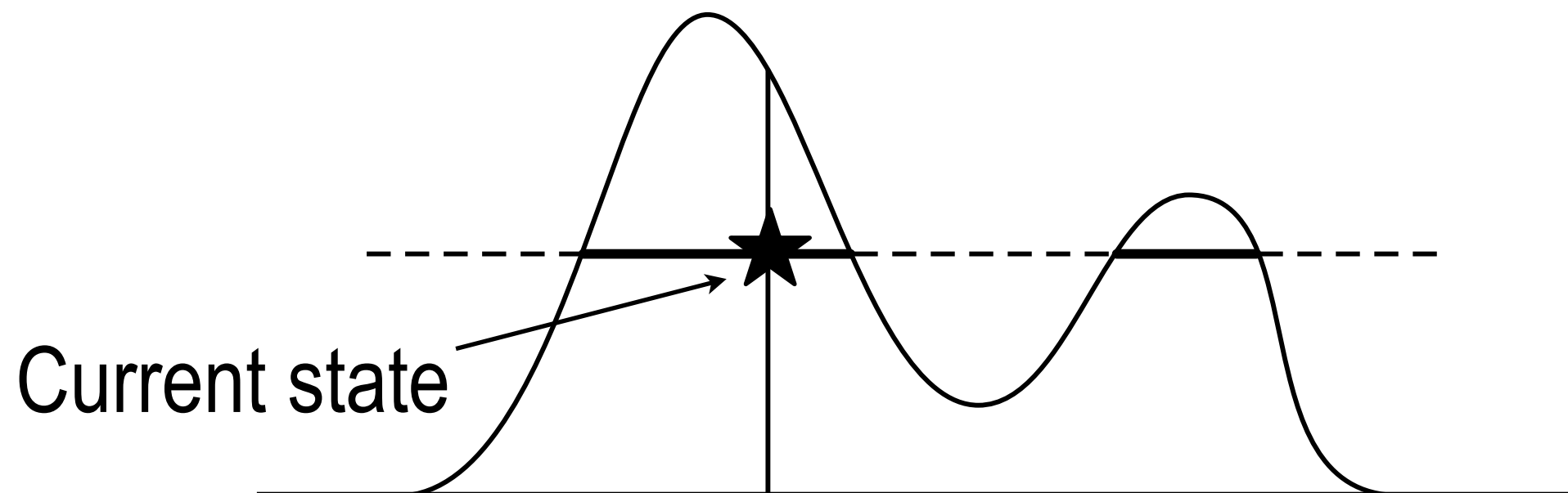
Example: *slice sampling*

Slice sampling

Goal: sampling from a r.v. X with density $f(x)/Z$, where Z is difficult to compute

Intuition: use a MCMC defined on the 2D space defined as the graph of the density

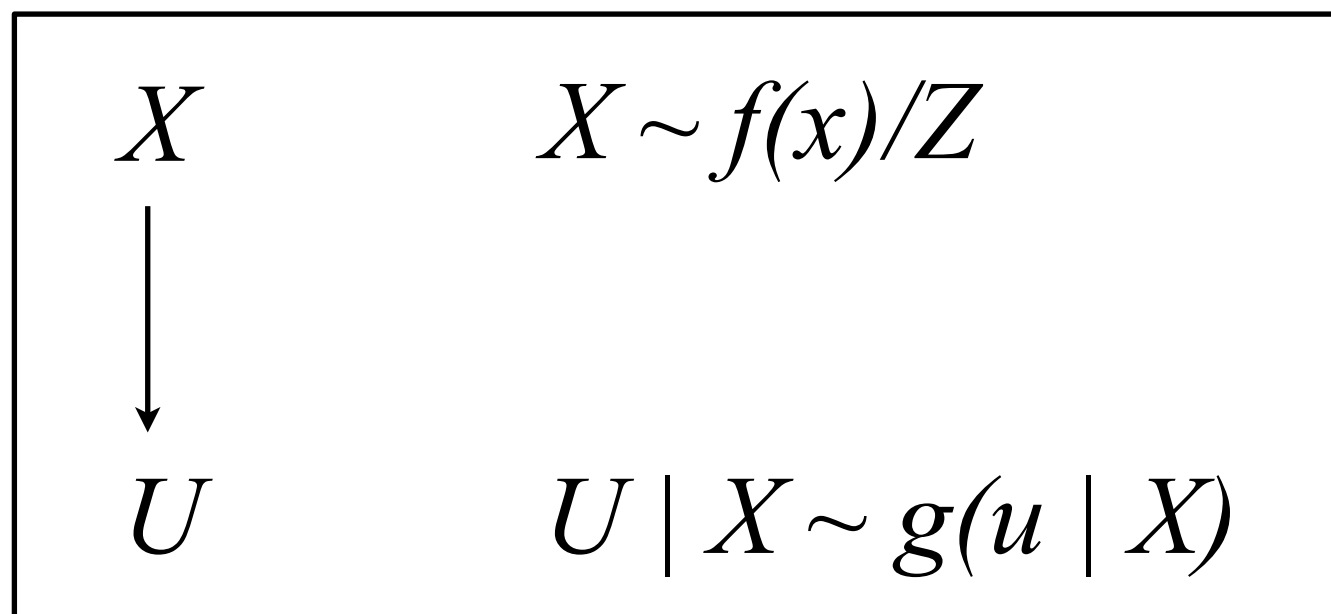
Moves: sample uniformly vertically or horizontally



Slice sampling

Goal: sampling from a r.v. X with density $f(x)/Z$, where Z is difficult to compute

General auxiliary variable construction: adding a new random variable U with the following graphical model does not change the marginal distribution of X , no matter what is the conditional density g of $U \mid X$

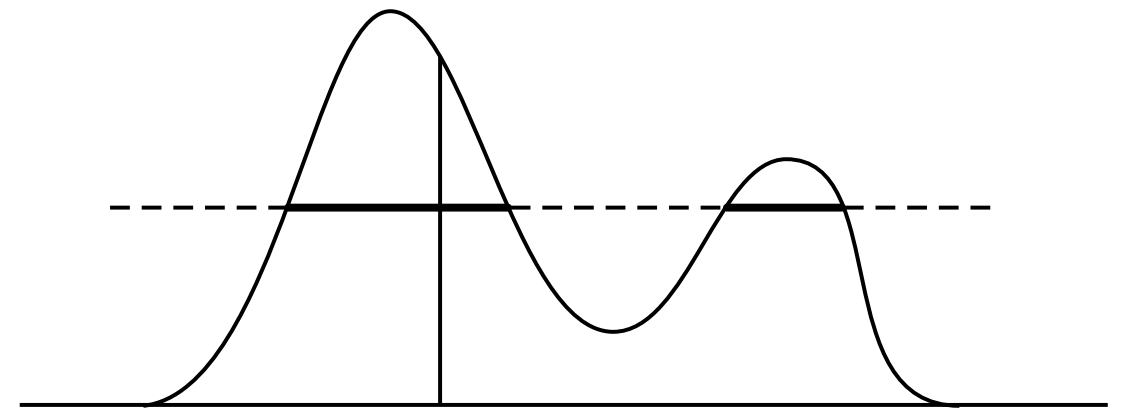


Slice sampler

$$X \quad X \sim f(x)/Z$$



$$U \quad U | X \sim \text{Uni}[0, f(X)]$$



Vertical move: $U | X \sim \text{Uni}[0, f(X)]$

Horizontal move: $X | U \sim \text{Uni}\{x : f(x) \geq U\}$

Note: Easier-to-compute alternatives to the horizontal move exist

Stopping criteria

Good: bound on the number of samples needed, or *exact (perfect) sampling* techniques

Bad (but somewhat useful): checking if the sequence of partial MC averages are approximately Cauchy; running independent, over-dispersed chains to check if the MC averages are close

Ugly: when the task is to compute a MC sum, heuristics based on staring at traceplots (value of $f(X_t)$ as a function of t), autocorrelation, and spectral density are misguided (see previous slide on LLN for Markov chains). Especially ugly when the heuristic is complicated to implement.

Perfect sampling

Idea 1: view MCMC algorithm as a deterministic maps T_v taking the current state x and an iid random uniform u : $T_v(x, u)$. The variable v selects the kernel and is also iid uni

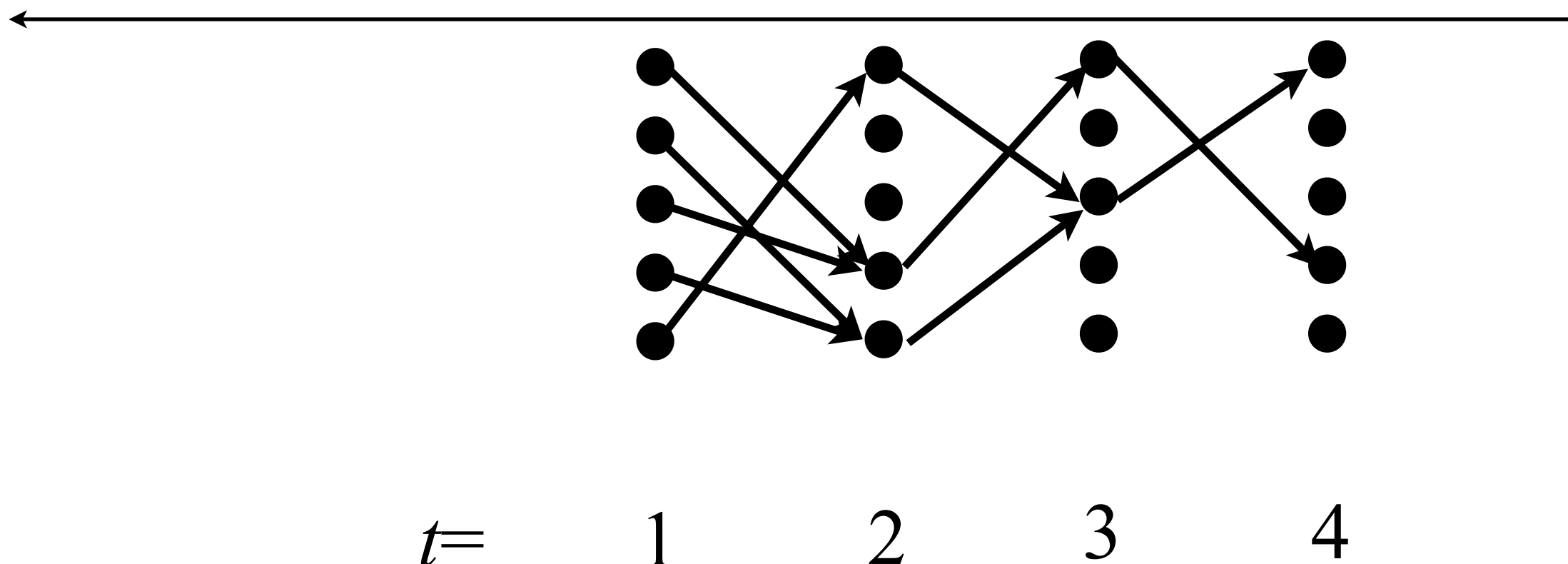
Idea 2: think backwards and generate an infinite sequence of past u, v

$$\begin{array}{cccc} T_{v_{-4}}(-, u_{-3}) & T_{v_{-2}}(-, u_{-2}) & T_{v_{-1}}(-, u_{-1}) & T_{v_0}(-, u_0) \\ \leftarrow \hline \end{array}$$

Perfect sampling

Idea 3: using the known u, v , find the possible states we can end up to in say 4 steps. Think about it as doing coupling, but instead of only 2 starting point, we consider all starting points simultaneously

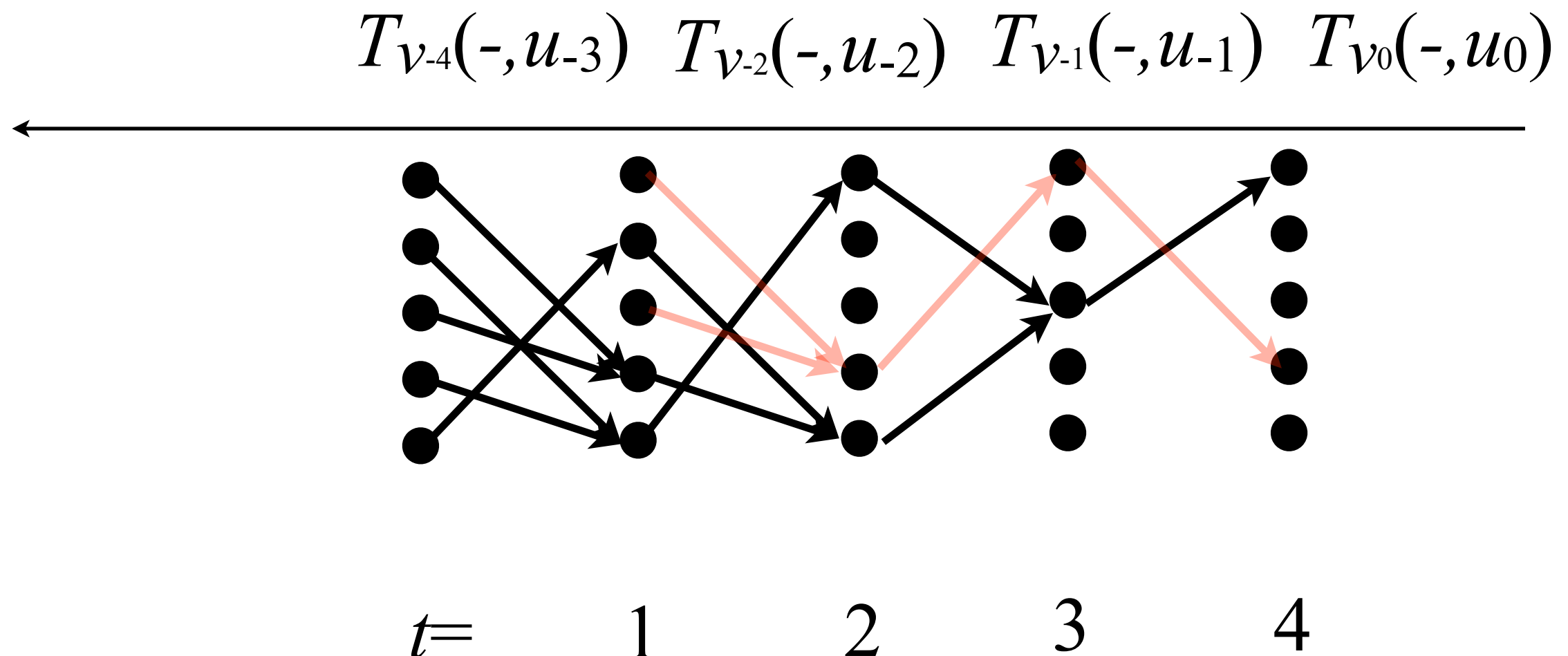
$$T_{v^{-4}}(-, u_{-3}) \quad T_{v^{-2}}(-, u_{-2}) \quad T_{v^{-1}}(-, u_{-1}) \quad T_{v^0}(-, u_0)$$



Perfect sampling

Idea 4: it that didn't rule out all the possible states at time 0, look further back in time to rule out more paths (using same randomness) until there is only one state at time zero.

The one state left is exactly distributed according to π !



Perfect sampling

In practice: too expensive to keep track of all states.

Solution: a partial order on the states such that

$$X_t \leq Y_t \Rightarrow X_{t+1} \leq Y_{t+1}$$

More precisely, such that for any fixed u, v :

$$x \leq y \Rightarrow T_v(x, u) \leq T_v(y, u)$$

Consequence: only need to keep track of maximal and minimal elements

Debugging MCMC algorithms

Important: Randomized algorithms are hard to implement

Test *all* small inputs: instead of a few big inputs. Either run the chain for a long time, or compute eigenvector explicitly and compare to true posterior.

Use synthetic data: how close do you get to generating parameters?
Is it improving when you generate more data?

Is the posterior calibrated? E.g. for binary variable, construct a histogram of % correct as a function of posterior of the prediction. There should be a linear trend.

Trick: fix random seeds to facilitate replication of bugs

Find the potential bugs

for $t = 1 \dots T$

Pick a kernel $q = q_\alpha$, $\alpha \sim M(x_{t-1})$

Loop....

1. Propose a new state x_{prop} according to $q(x | x_{t-1})$

2. Compute:

$$A(x_{t-1} \rightarrow x_{\text{prop}}) = \min \left\{ 1, \frac{\text{target}(x_{t-1})q(x_{\text{prop}} | x_{t-1})}{\text{target}(x_{\text{prop}})q(x_{t-1} | x_{\text{prop}})} \right\}$$

3. Generate a Unif[0,1] number u

.... while $u > A(x_{t-1} \rightarrow x_{\text{prop}})$

Set x_t to x_{prop}

Find the potential bugs

for $t = 1 \dots T$

Pick a kernel $q = q_\alpha, \alpha \sim M(x_{t-1})$

Loop....

1. Propose a new state x_{prop} according to $q(x | x_{t-1})$

2. Compute:

$$A(x_{t-1} \rightarrow x_{\text{prop}}) = \min \left\{ 1, \frac{\text{target}(x_{t-1})q(x_{\text{prop}} | x_{t-1})}{\text{target}(x_{\text{prop}})q(x_{t-1} | x_{\text{prop}})} \right\}$$

3. Generate a Unif[0,1] number u

.... while $u > A(x_{t-1} \rightarrow x_{\text{prop}})$

Set x_t to x_{prop}

- Ratio is upside down !
- Mixing of kernels
distribution should not
depend on x
- No while loop !
(c.f. rejection sampling)

Find the potential bug



Binary r.v.s

```
 $S_X, S_Y, S_Z = 0$   
 $N_X, N_Y, N_Z = 0$   
for  $t = 1 \dots T$   
  if  $t$  is Odd  
     $(X, Y) \sim \text{Block Gibbs}$   
     $S_X = S_X + X; N_X = S_X + 1$   
     $S_Y = S_Y + Y; N_Y = S_Y + 1$   
  else  
    [do the same for  $(Y, Z)$ ]  
return  $S_X/N_X, S_Y/N_Y, S_Z/N_Z$ 
```

Find the potential bug



Binary r.v.s

$S_X, S_Y, S_Z = 0$

$N_X, N_Y, N_Z = 0$

for $t = 1 \dots T$

 if t is Odd

$(X, Y) \sim \text{Block Gibbs}$

$S_X = S_X + X; N_X = S_X + 1$

$S_Y = S_Y + Y; N_Y = S_Y + 1$

 else

 [do the same for (Y, Z)]

return $S_X/N_X, S_Y/N_Y, S_Z/N_Z$

The partial sum S_Y/N_Y gets updated more often (it's ok to update random variables different numbers of times, but all partial sums need to be updated at each iteration---even when the corresponding r.v. doesn't change)

More on the previous bug

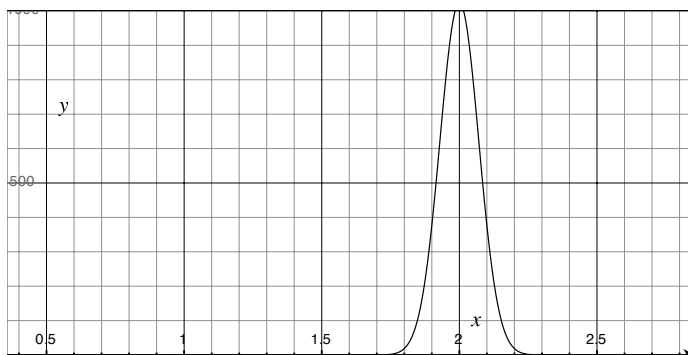
In Gibbs sampling, we are updating only a small number of variables, it seems silly to update everything...

When there is no overlap between the sampled blocks, and every block is asymptotically sampled the same number of times, can update only what changed

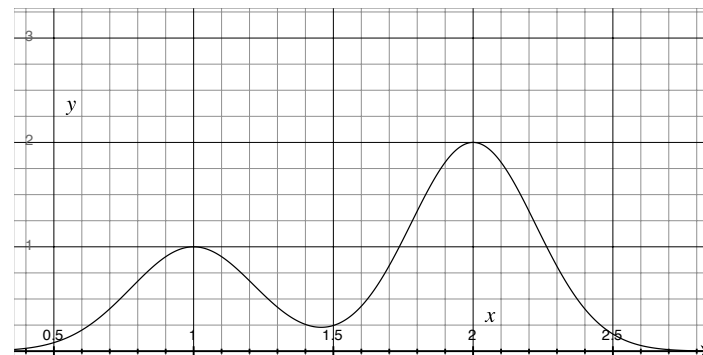
Otherwise, can use a delayed update datastructure (keep track of the last MCMC iteration each variable/coordinate is updated, and when it gets updated again, add delta time multiplied by last value.

Other heuristics

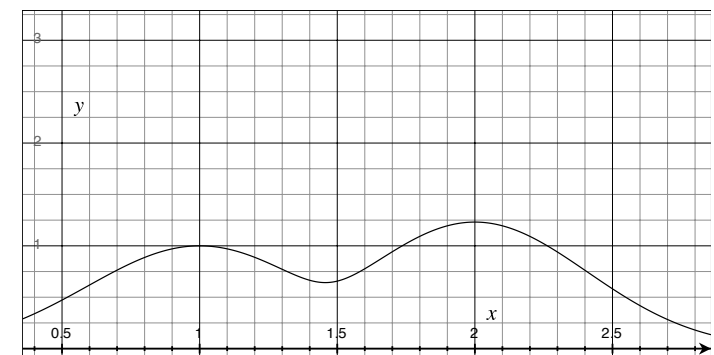
Simulated Annealing: exponentiation of the target distribution



Cold (why?):
 $P(X=x)=f(x)^{10}$



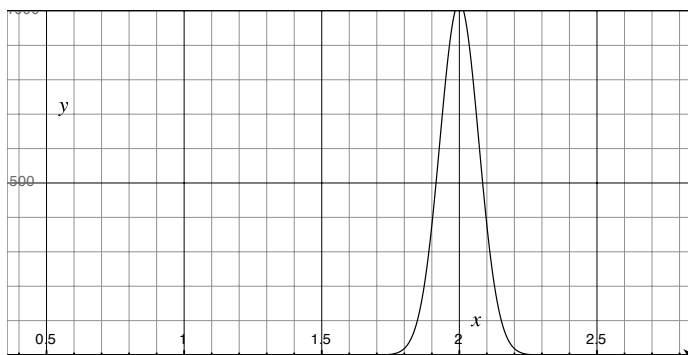
Room temperature:
 $P(X=x)=f(x)$



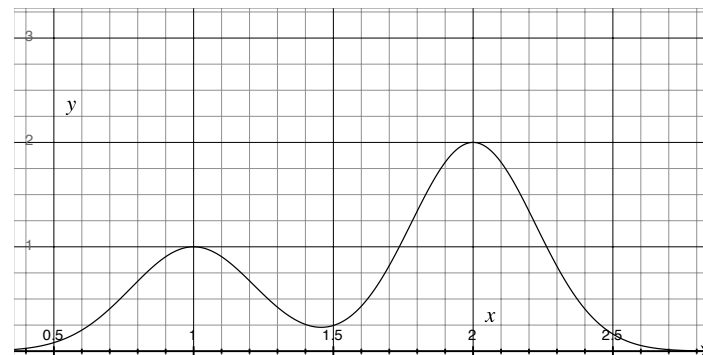
Hot (why?):
 $P(X=x)=f(x)^{0.3}$

Other heuristics

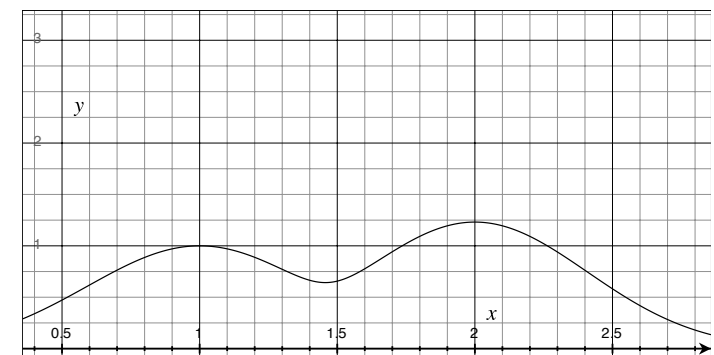
Simulated Annealing: exponentiation of the target distribution



Cold (why?):
 $P(X=x)=f(x)^{1.0}$



Room temperature:
 $P(X=x)=f(x)$

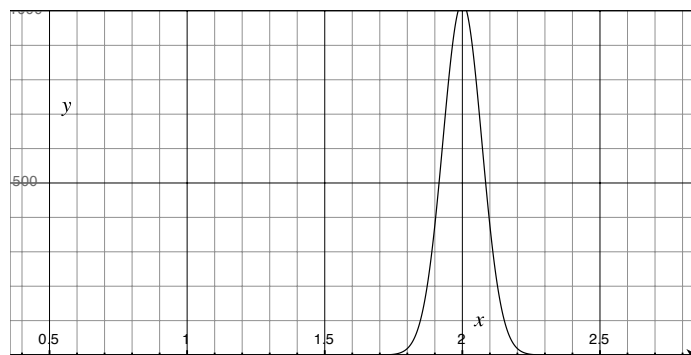


Hot (why?):
 $P(X=x)=f(x)^{0.3}$

To search a
maximum (MAP
configuration)

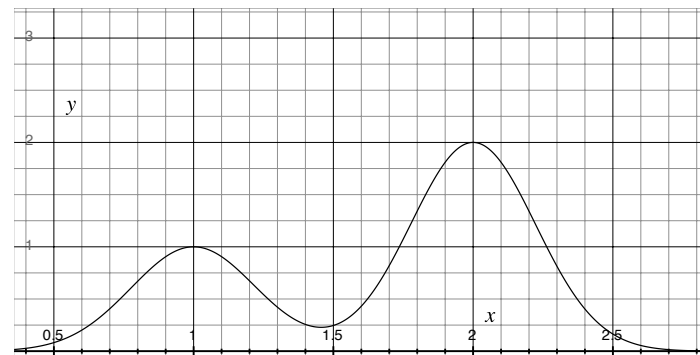
Other heuristics

Simulated Annealing: exponentiation of the target distribution

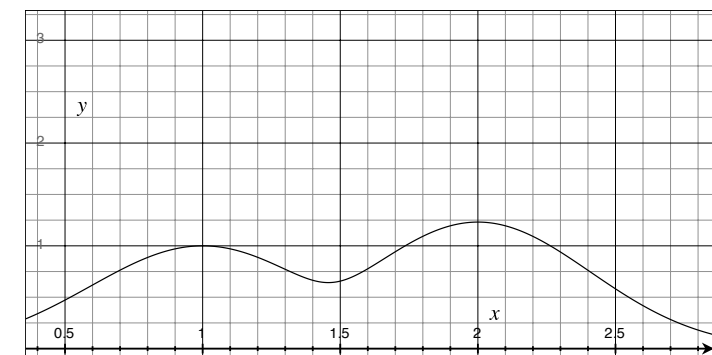


Cold (why?):
 $P(X=x)=f(x)^{1.0}$

To search a
maximum (MAP
configuration)



Room temperature:
 $P(X=x)=f(x)$

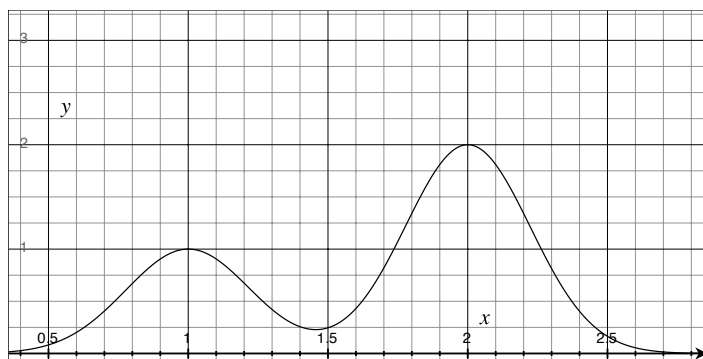


Hot (why?):
 $P(X=x)=f(x)^{0.3}$

To make it easier
to jump from one
mode to the other

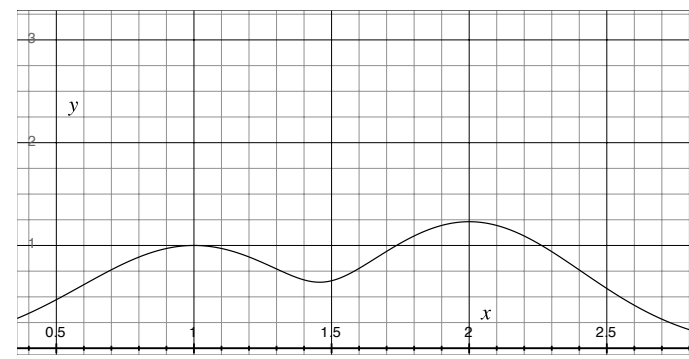
Other heuristics

Tempering: expand the state space to one independent extra chain, with higher temperature. States $x = (x^{(1)}, x^{(2)})$



Main chain at room temp.

$$P(X = x^{(1)}) = f(x^{(1)})$$



Hot copy

$$P(X = x^{(2)}) = f(x^{(2)})^{0.3}$$

Swap move: introduce a proposal distribution that swaps the current states in the two chains.

MH ratio:

$$\frac{\mathbb{P}(x_{\text{prop}})}{\mathbb{P}(x_{t-1})} = \frac{f(x^{(2)}) (f(x^{(1)}))^{0.3}}{f(x^{(1)}) (f(x^{(2)}))^{0.3}} = \frac{(f(x^{(2)}))^{0.7}}{(f(x^{(1)}))^{0.7}}$$

Other heuristics

Tempering: general version is a chain of chains

$$x = (x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(K)})$$

of increasing temperatures.

Moves: swaps between each pair of chain (ratio has the same form but with different exponents)

Countably infinite chain

Danger: even when the chain is irreducible, in infinite state spaces, there is a risk that the chain never comes back to its initial point

Example: 'A drunk man will eventually find his way home, but a drunk bird may get lost forever'

Formalization of this joke...

Consequences on the asymptotic theory

All the the theory is salvaged if we assume *positive recurrence*

I.e., we have our theorem on existence of stationary distribution, convergence of X_t to the stationary distribution, and convergence of Monte Carlo averages)

Continuous state space

Idea: Assume a base measure ν and use the same definition as before, but for set A with $\nu(A) > 0$.

Example (*Harris recurrence*): For all A with $\nu(A) > 0$, the set A is visited infinitely often with probability one

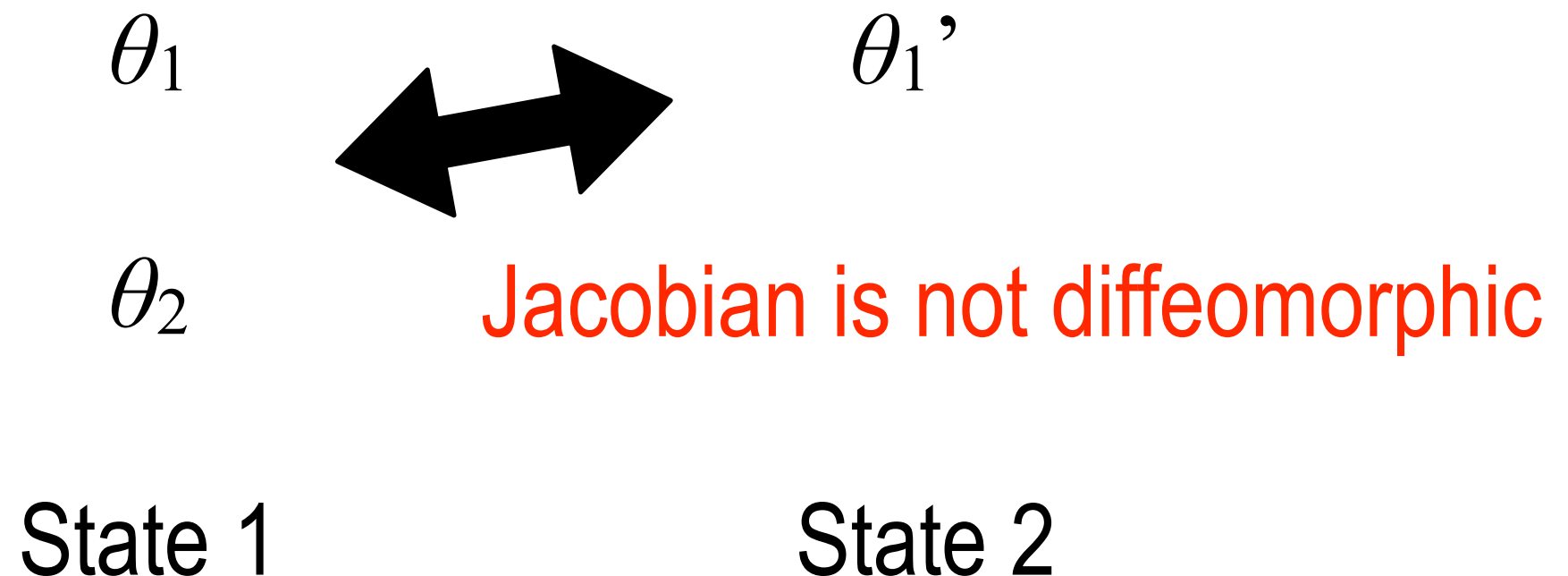
For most samplers theory goes through easily (only need to compute an extra Jacobian).

One useful trick not needed in discrete spaces:
Reversible Jump MCMC

Reversible Jump MCMC

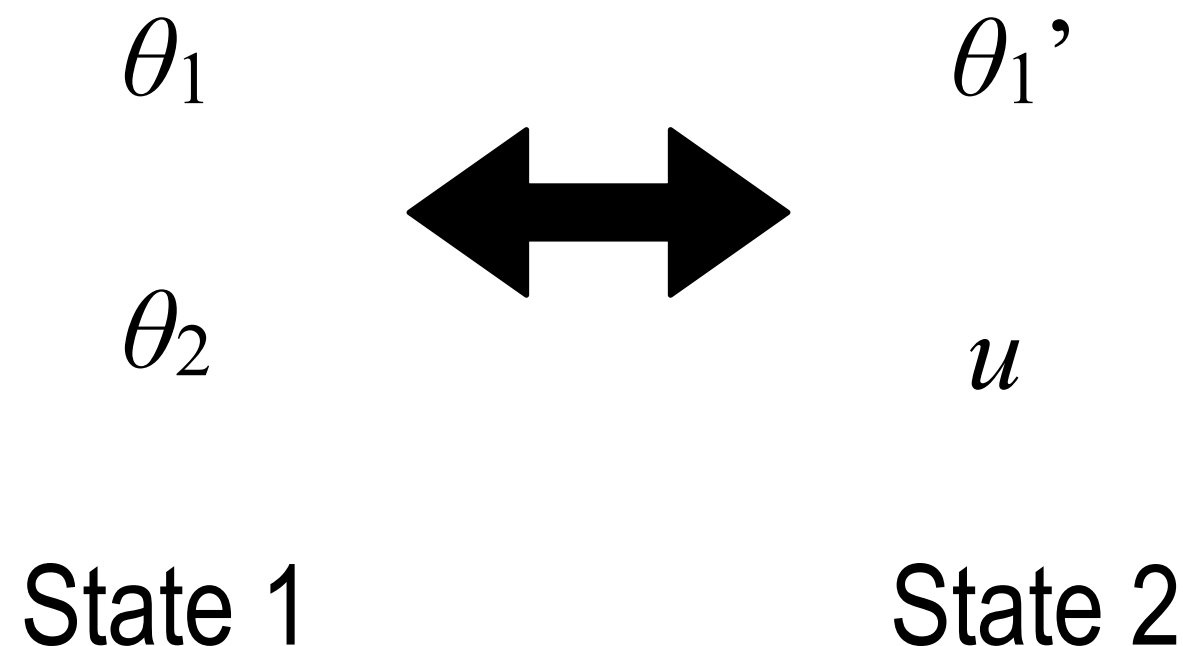
Goal: sample a continuous space with unknown number of dimensions. E.g.: model selection--not sure if we should use a model with one or two parameters.

Problem:



Reversible Jump MCMC

Solution: introduce iid auxiliary variables u to make each state of the same dimensionality (don't actually need to represent them across iterations)



Variational inference

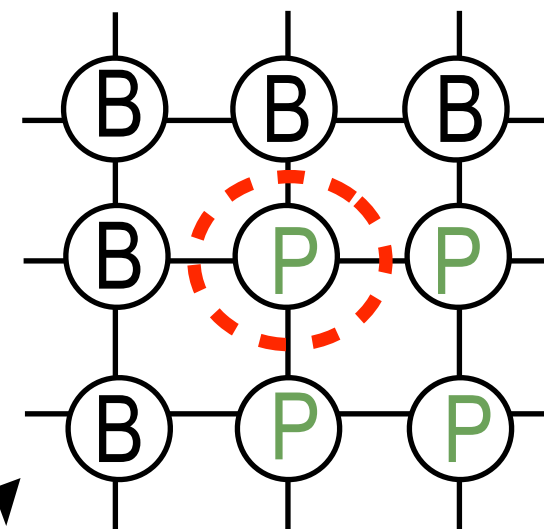
Road map

Hard probabilistic
inference problems

$$\text{target}(x) = \mathbb{P}(X = x | \text{obs, params})$$

1

Deterministic
algorithms



2

3

$$\lambda_{s_1}(A) = \lambda_{s_1, s_2}(A, \mathbf{R}) \quad [\text{marginalization}]$$

$$\lambda_{s_1, s_2}(A_1, A_2) = \lambda_{s_2, s_1}(A_2, A_1)$$

Probabilistic inference as
an optimization problem

Quick review of exponential family

Sufficient statistic

Parameter

$$\mathbb{P}(\mathbf{X}_{\boldsymbol{\theta}} \in B) = \sum_{x \in B} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle - A(\boldsymbol{\theta})\} \nu(x),$$

$$A(\boldsymbol{\theta}) = \log \sum_{x \in \mathcal{X}} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle\} \nu(x),$$

A counting
measure

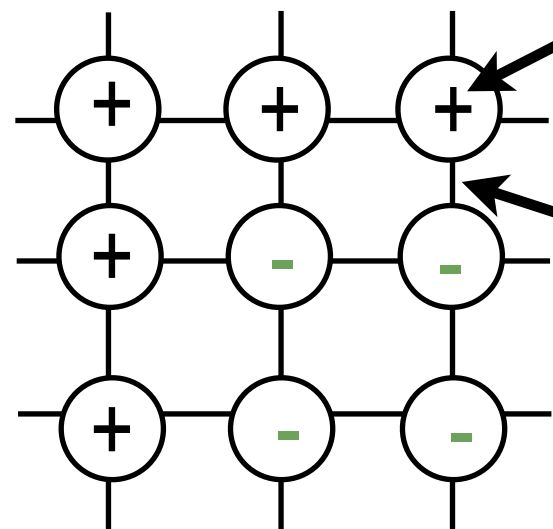
Log partition function

Large discrete set
(e.g. all configs of an Ising model)

Example of sufficient statistics

Ising model

One node



Pairs of nodes

$\phi(x) =$

$$\begin{bmatrix} \mathbf{1}[x_{1,1} = +] \\ \mathbf{1}[x_{1,1} = -] \\ \mathbf{1}[x_{1,2} = +] \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{1}[x_{1,1} = +, x_{1,2} = +] \\ \mathbf{1}[x_{1,1} = +, x_{1,2} = -] \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} \theta_{1,1,+} \\ \theta_{1,1,-} \\ \theta_{1,2,+} \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} \theta_{1,1,+;1,2,+} \\ \theta_{1,1,+;1,2,-} \\ \vdots \end{bmatrix}$$

D

‘Over-complete’ sufficient statistic

What we are trying to compute

Moments:

$$\mu = \mathbb{E}[\phi(X)] = \begin{bmatrix} \mu_{1,1,+} \\ \mu_{1,1,-} \\ \mu_{1,2,+} \\ \vdots \\ \mu_{1,1,+;1,2,+} \\ \mu_{1,1,+;1,2,-} \\ \vdots \end{bmatrix} \begin{bmatrix} \mathbf{1}[x_{1,1} = +] \\ \mathbf{1}[x_{1,1} = -] \\ \mathbf{1}[x_{1,2} = +] \\ \vdots \\ \mathbf{1}[x_{1,1} = +, x_{1,2} = +] \\ \mathbf{1}[x_{1,1} = +, x_{1,2} = -] \\ \vdots \end{bmatrix} \begin{bmatrix} \theta_{1,1,+} \\ \theta_{1,1,-} \\ \theta_{1,2,+} \\ \vdots \\ \theta_{1,1,+;1,2,+} \\ \theta_{1,1,+;1,2,-} \\ \vdots \end{bmatrix}$$

D

and *log partition function*: $A(\boldsymbol{\theta}) = \log \sum_{x \in \mathcal{X}} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle\} \nu(x)$

Important properties

The gradient of the log partition function is equal to the moments:

$$\nabla A(\boldsymbol{\theta}) = \mathbb{E}[\boldsymbol{\phi}(\mathbf{X}_{\boldsymbol{\theta}})]$$

The hessian of the log partition function is equal to the covariance matrix:

$$H(A(\boldsymbol{\theta})) = \mathbf{V}ar[\boldsymbol{\phi}(\mathbf{X}_{\boldsymbol{\theta}})].$$

Consequence: A is a convex function