Sampling Algorithms

In general

- A **sampling algorithm** is an algorithm that outputs samples $x_1, x_2, \ldots$ from a given distribution P or density p.
- Sampling algorithms can for example be used to approximate expectations:

$$\mathbb{E}_p[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

Inference in Bayesian models

Suppose we work with a Bayesian model whose posterior $\hat{Q}_n := Q[d\theta|X_{1:n}]$ cannot be computed analytically.

- We will see that it can still be possible to *sample* from $\hat{Q}_n$.
- Doing so, we obtain samples $\theta_1, \theta_2, \ldots$ distributed according to $\hat{Q}_n$.
- This reduces posterior estimation to a density estimation problem (i.e. estimate $\hat{Q}_n$ from $\theta_1, \theta_2, \ldots$).

Posterior expectations

If we are only interested in some statistic of the posterior of the form $\mathbb{E}_{\hat{Q}_n}[f(\Theta)]$ (e.g. the posterior mean), we can again approximate by

$$\mathbb{E}_{\hat{Q}_n}[f(\Theta)] \approx \frac{1}{m}\sum_{i=1}^{m} f(\theta_i) \ .$$

Example: Predictive distribution

The **posterior predictive distribution** is our best guess of what the next data point x_{n+1} looks like, given the posterior under previous observations:

$$P[dx_{n+1}|x_{1:n}] := \int_{\mathbf{T}} \mathbf{p}(dx_{n+1}|\theta) Q[d\theta|X_{1:n} = x_{1:n}] \ .$$

This is one of the key quantities of interest in Bayesian statistics.

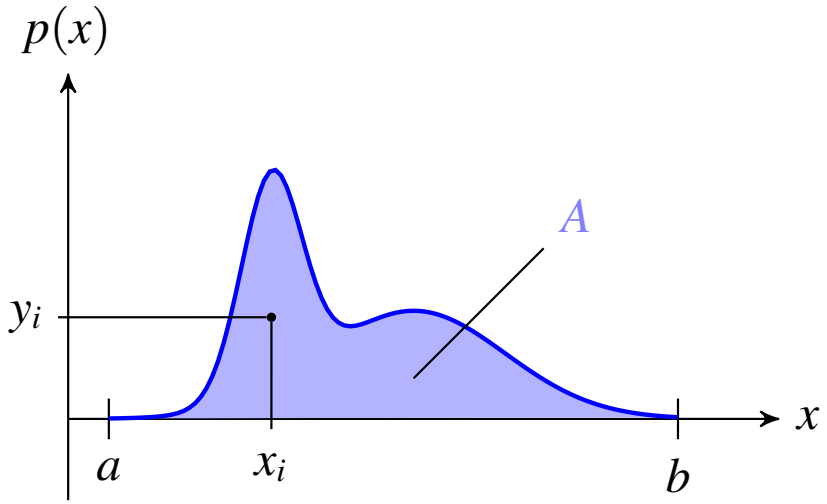
Computation from samples

The predictive is a posterior expectation, and can be approximated as a sample average:

$$p(x_{n+1}|x_{1:n}) = \mathbb{E}_{\hat{Q}_n}[p(x_{n+1}|\Theta)] \approx \frac{1}{m}\sum_{i=1}^{m} p(x_{n+1}|\theta_i)$$

Basic Sampling: Area Under Curve

Say we are interested in a probability density p on the interval $[a, b]$.



Key observation

Suppose we can define a uniform distribution U_A on the blue area A under the curve. If we sample

$$(x_1, y_1), (x_2, y_2), \ldots \sim_{\text{iid}} U_A$$

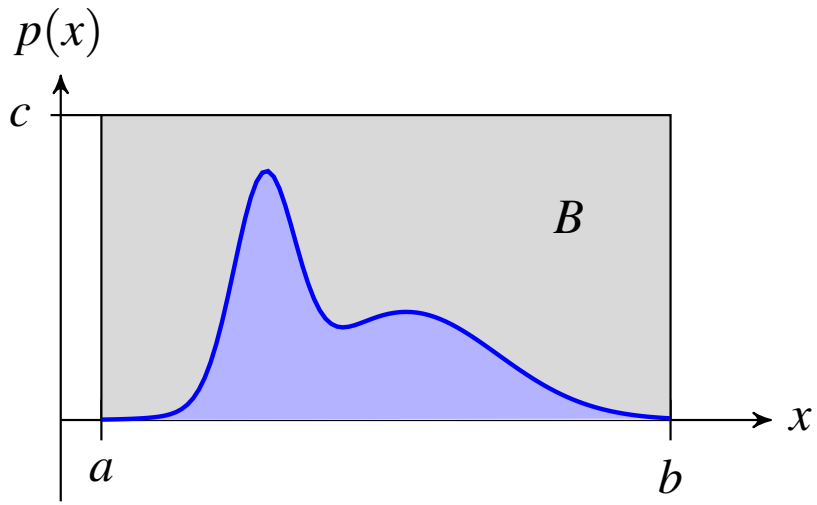
and discard the vertical coordinates y_i, **the x_i are distributed according to p**,

$$x_1, x_2, \ldots \sim_{\text{iid}} p \ .$$

Problem: Defining a uniform distribution is easy on a rectangular area, but difficult on an arbritrarily shaped one.

Solution: Rejection sampling

We can enclose p in box, and sample uniformly from the box B.



- We can sample (x_i, y_i) uniformly on B by sampling

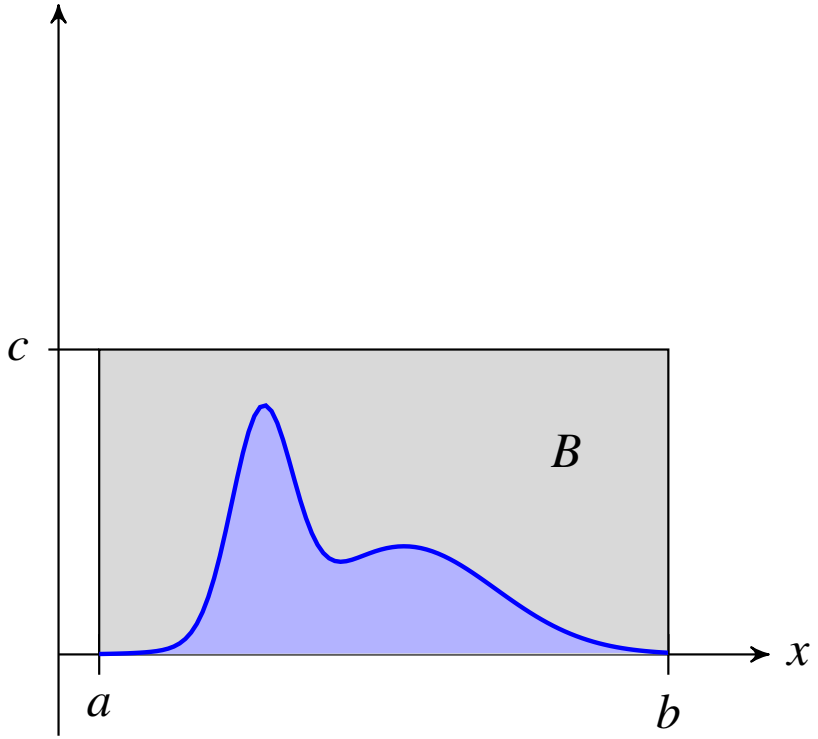
$$x_i \sim \text{Uniform}[a, b] \qquad \text{and} \qquad y_i \sim \text{Uniform}[0, c] \ .$$

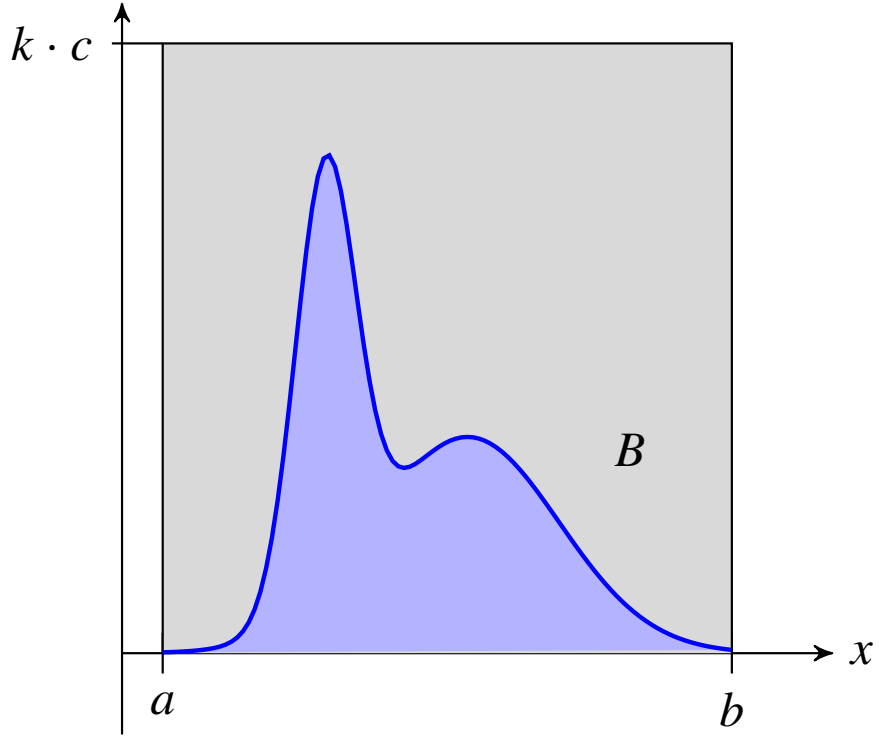
- If $(x_i, y_i) \in A$ (that is: if $y_i \leq p(x_i)$), keep the sample.
  Otherwise: discard it ("reject" it).

Result: The remaining (non-rejected) samples are uniformly distributed on A.

SCALING

This strategy still works if we scale the vertically by some constant $k > 0$:





We simply sample $y_i \sim \text{Uniform}[0, kc]$ instead of $y_i \sim \text{Uniform}[0, c]$.

Consequence

For sampling, it is sufficient if p is known only up to normalization (i.e. if only the shape of p is known).

Sampling methods usually assume that we can evaluate the target distribution p up to a constant. That is:

$$p(x) \quad = \quad \frac{1}{\tilde{Z}}\,\tilde{p}(x)\ ,$$

and we can compute $\tilde{p}(x)$ for any given x, but we do not know $\tilde{Z}$.

We have to pause for a moment and convince ourselves that there are useful examples where this assumption holds.

Example 1: Simple posterior

For an arbitrary posterior computed with Bayes' theorem, we could write

$$\Pi(\theta|x_{1:n}) = \frac{\prod_{i=1}^{n} p(x_i|\theta)q(\theta)}{\tilde{Z}} \qquad \text{with} \qquad \tilde{Z} = \int_{\mathbf{T}} \prod_{i=1}^{n} p(x_i|\theta)q(\theta)d\theta\ .$$

Provided that we can compute the numerator, we can sample without computing the normalization integral $\tilde{Z}$.

Example 2: Bayesian Mixture Model

Recall that the posterior of the BMM is (up to normalization):

$$\hat{q}_n(c_{1:K}, \theta_{1:K} | x_{1:n}) \propto \prod_{i=1}^{n} \Big(\sum_{k=1}^{K} c_k p(x_i|\theta_k) \Big) \Big(\prod_{k=1}^{K} q(\theta_k|\lambda, y) \Big) q_{\text{Dirichlet}}(c_{1:K})$$

We already know that we can discard the normalization constant, but can we evaluate the non-normalized posterior $\tilde{q}_n$?

- The problem with computing $\tilde{q}_n$ (as a function of unknowns) is that the term $\prod_{i=1}^{n} \left(\sum_{k=1}^{K} \dots \right)$ blows up into K^n individual terms.
- If we *evaluate* $\tilde{q}_n$ for specific values of c, x and θ, $\sum_{k=1}^{K} c_k p(x_i|\theta_k)$ collapses to a single number for each x_i, and we just have to multiply those n numbers.

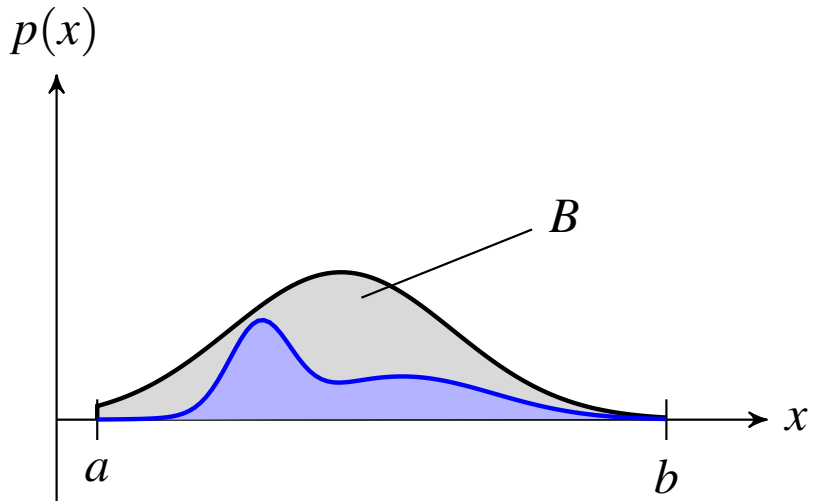
So: Computing $\tilde{q}_n$ as a formula in terms of unknowns is difficult; evaluating it for specific values of the arguments is easy.

Rejection Sampling on $\mathbb{R}^d$

If we are not on the interval, sampling uniformly from an enclosing box is not possible (since there is no uniform distribution on all of $\mathbb{R}$ or $\mathbb{R}^d$).

Solution: Proposal density

Instead of a box, we use *another distribution* r to enclose p:

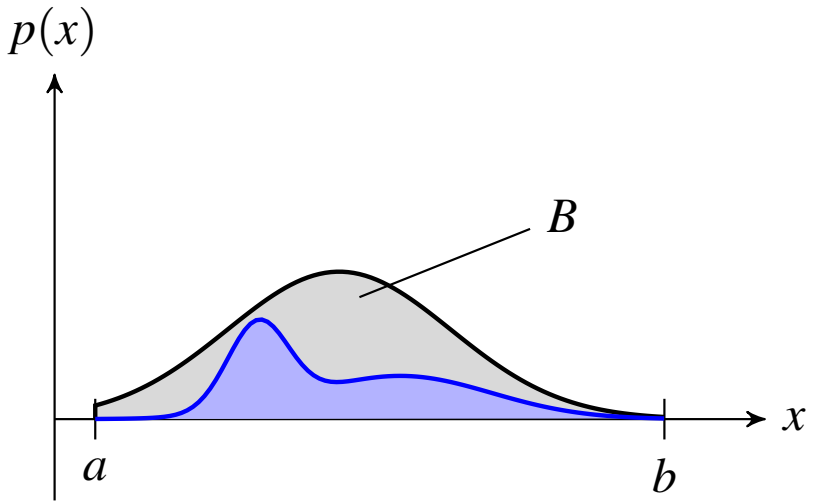


To generate B under r, we apply similar logic as before backwards:

- Sample $x_i \sim r$.
- Sample $y_i \sim \text{Uniform}[0, r(x_i)]$.

r is always a simple distribution which we can sample and evaluate.

Rejection Sampling on $\mathbb{R}^d$



- Choose a simple distribution r from which we know how to sample.
- Scale $\tilde{p}$ such that $\tilde{p}(x) < r(x)$ everywhere.
- Sampling: For $i = 1, 2, \ldots,$:
  1. Sample $x_i \sim r$.
  2. Sample $y_i \sim \text{Uniform}[0, r(x_i)]$.
  3. If $y_i < \tilde{p}(x_i)$, keep x_i.
  4. Else, discard x_i and start again at (1).
- The surviving samples $x_1, x_2, \ldots$ are distributed according to p.

If we draw proposal samples x_i i.i.d. from r, the resulting sequence of accepted samples produced by rejection sampling is again i.i.d. with distribution p. Hence:

Rejection samplers produce i.i.d. sequences of samples.

Important consequence

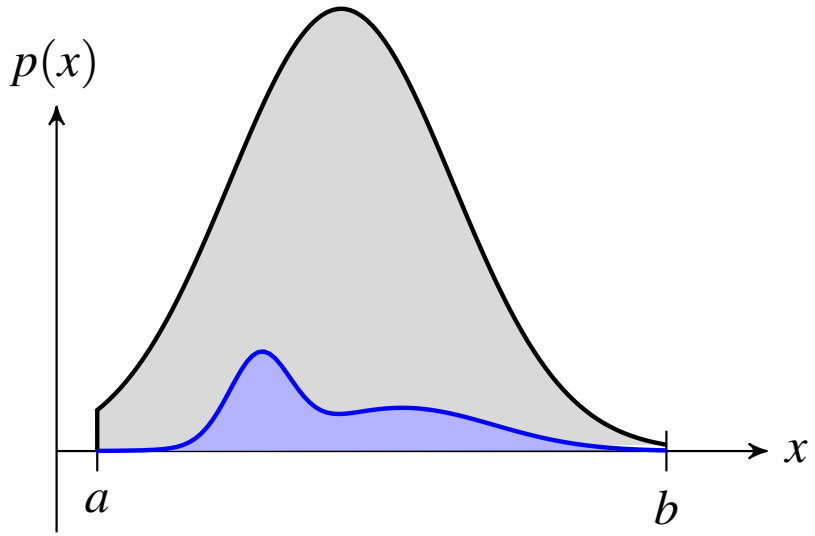
If samples $x_1, x_2, \ldots$ are drawn by a rejection sampler, the sample average

$$\frac{1}{m}\sum_{i=1}^{m} f(x_i)$$

(for some function f) is an unbiased estimate of the expectation $\mathbb{E}_p[f(X)]$.

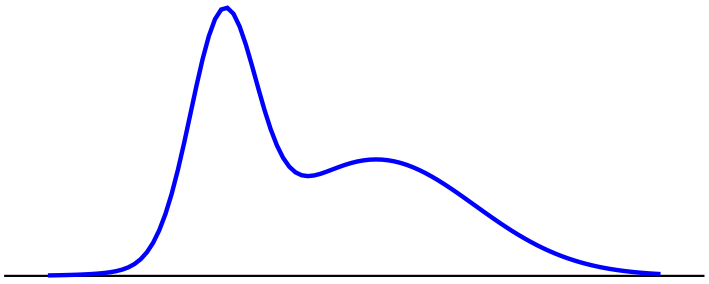
Efficiency

The fraction of accepted samples is the ratio $\frac{|A|}{|B|}$ of the areas under the curves $\tilde{p}$ and r.

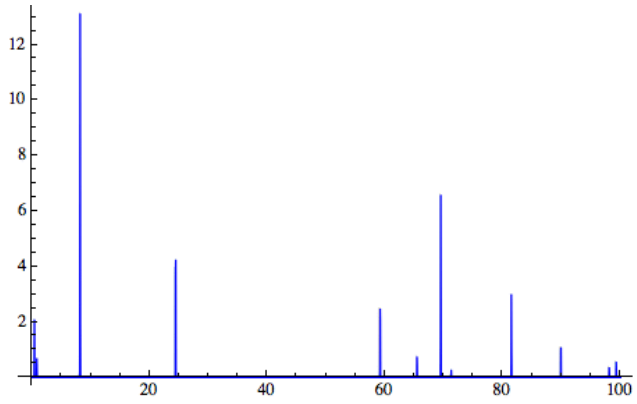


If r is not a reasonably close approximation of p, we will end up rejecting a lot of proposal samples.

An important bit of imprecise intuition



Example figures for sampling methods tend to look like this.



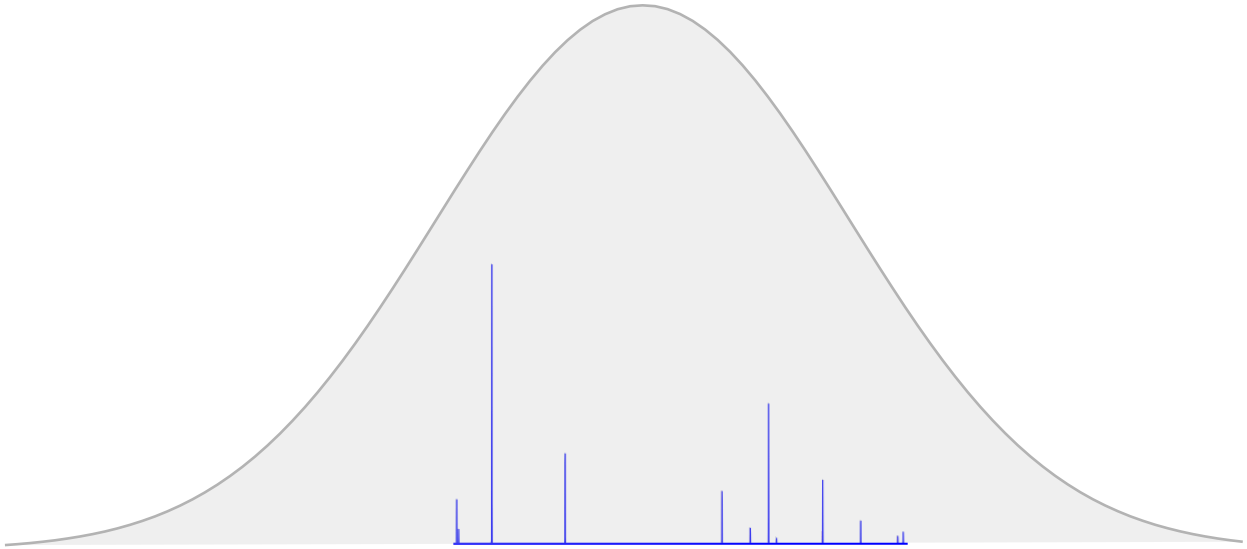
A high-dimensional distribution of correlated RVs will look rather more like this.

Sampling is usually used in multiple dimensions. Reason, roughly speaking:

- Intractable posterior distributions arise when there are several *interacting* random variables. The interactions make the joint distribution complicated.
- In one-dimensional problems (1 RV), we can usually compute the posterior analytically.
- Independent multi-dimensional distributions factorize and reduce to one-dimensional case.

Warning: Never (!!!) use sampling if you can solve analytically.

WHY IS NOT EVERY SAMPLER A REJECTION SAMPLER?



We can easily end up in situations where we accept only one in 10^6 (or 10^{10}, or 10^{20},...) proposal samples. Especially in higher dimensions, we have to expect this to be not the exception but the rule.

The rejection problem can be fixed easily if we are only interested in approximating an expectation $\mathbb{E}_p[f(X)]$.

Simple case: We can evaluate p

Suppose p is the target density and q a proposal density. An expectation under p can be rewritten as

$$\mathbb{E}_p[f(X)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_q\left[\frac{f(X)p(X)}{q(X)}\right]$$

Importance sampling

We can sample $x_1, x_2, \ldots$ from q and approximate $\mathbb{E}_p[f(X)]$ as

$$\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(x_i)\frac{p(x_i)}{q(x_i)}$$

There is no rejection step; all samples are used.

This method is called **importance sampling**. The coefficients $\frac{p(x_i)}{q(x_i)}$ are called **importance weights**.

General case: We can only evaluate $\tilde{p}$

In the general case,

$$p = \frac{1}{Z_p}\tilde{p} \qquad \text{and} \qquad q = \frac{1}{Z_q}\tilde{q}\ ,$$

and Z_p (and possibly Z_q) are unknown. We can write $\frac{Z_p}{Z_q}$ as

$$\frac{Z_p}{Z_q} = \frac{\int \tilde{p}(x)dx}{Z_q} = \frac{\int \tilde{p}(x)\frac{q(x)}{q(x)}dx}{Z_q} = \int \tilde{p}(x)\frac{q(x)}{Z_q \cdot q(x)}dx = \mathbb{E}_q\left[\frac{\tilde{p}(X)}{\tilde{q}(X)}\right]$$

Approximating the constants

The fraction $\frac{Z_p}{Z_q}$ can be approximated using samples $x_{1:m}$ from q:

$$\frac{Z_p}{Z_q} = \mathbb{E}_q\left[\frac{\tilde{p}(X)}{\tilde{q}(X)}\right] \approx \frac{1}{m}\sum_{i=1}^{m}\frac{\tilde{p}(x_i)}{\tilde{q}(x_i)}$$

Approximating $\mathbb{E}_p[f(X)]$

$$\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(x_i)\frac{p(x_i)}{q(x_i)} = \frac{1}{m}\sum_{i=1}^{m} f(x_i)\frac{Z_q}{Z_p}\frac{\tilde{p}(x_i)}{\tilde{q}(x_i)} = \sum_{i=1}^{m}\frac{f(x_i)\frac{\tilde{p}(x_i)}{\tilde{q}(x_i)}}{\sum_{i=j}^{m}\frac{\tilde{p}(x_j)}{\tilde{q}(x_j)}}$$

Conditions

- Given are a target distribution p and a proposal distribution q.
- $p = \frac{1}{Z_p}\tilde{p}$ and $q = \frac{1}{Z_q}\tilde{q}$.
- We can evaluate $\tilde{p}$ and $\tilde{q}$, and we can sample q.
- The objective is to compute $\mathbb{E}_p[f(X)]$ for a given function f.

Algorithm

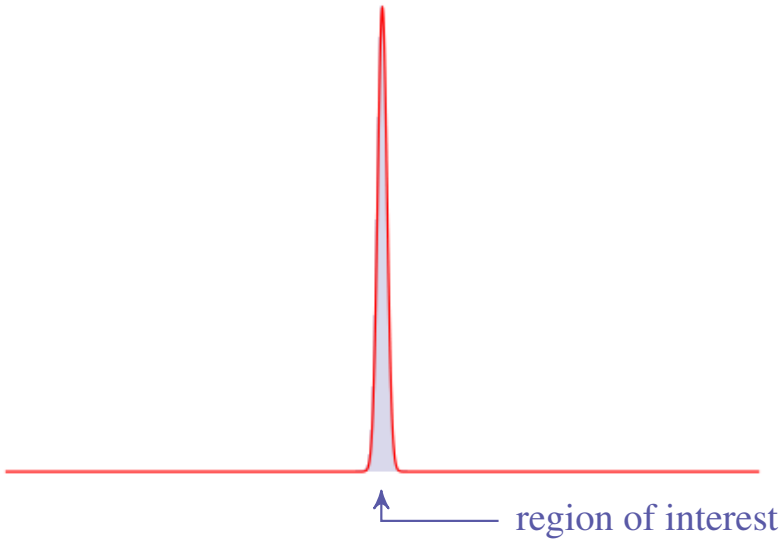
1. Sample $x_1, \ldots, x_m$ from q.
2. Approximate $\mathbb{E}_p[f(X)]$ as

$$\mathbb{E}_p[f(X)] \approx \frac{\sum_{i=1}^{m} f(x_i)\frac{\tilde{p}(x_i)}{\tilde{q}(x_i)}}{\sum_{i=j}^{m} \frac{\tilde{p}(x_j)}{\tilde{q}(x_j)}}$$

Markov Chain Monte Carlo

MOTIVATION

Suppose we rejection-sample a distribution like this:



Once we have drawn a sample in the narrow region of interest, we would like to continue drawing samples within the same region. That is only possible if each sample *depends on the location of the previous sample*.

Proposals in rejection sampling are i.i.d. Hence, once we have found the region where p concentrates, we forget about it for the next sample.

Recall: Markov chain

- A sufficiently nice Markov chain (MC) has an invariant distribution P_{inv}.
- Once the MC has converged to P_{inv}, each sample x_i from the chain has marginal distribution P_{inv}.

Markov chain Monte Carlo

We want to sample from a distribution with density p. Suppose we can define a MC with invariant distribution $P_{\text{inv}} \equiv p$. If we sample $x_1, x_2, \ldots$ from the chain, then once it has converged, we obtain samples

$$x_i \sim p \ .$$

This sampling technique is called **Markov chain Monte Carlo** (**MCMC**).

Note: For a Markov chain, x_{i+1} can depend on x_i, so at least in principle, it is possible for an MCMC sampler to "remember" the previous step and remain in a high-probability location.

Continuous Markov Chain

For MCMC, state space now has to be the domain of p, so we often need to work with continuous state spaces.

Continuous Markov chain

A continuous Markov chain is defined by an initial distribution P_{init} and conditional probability $\mathbf{t}(dy|x)$, the **transition probability** or **transition kernel**.

In the discrete case, $\mathbf{t}(y = i|x = j)$ is the entry $\mathbf{T}_{ij}$ of the transition matrix $\mathbf{T}$.

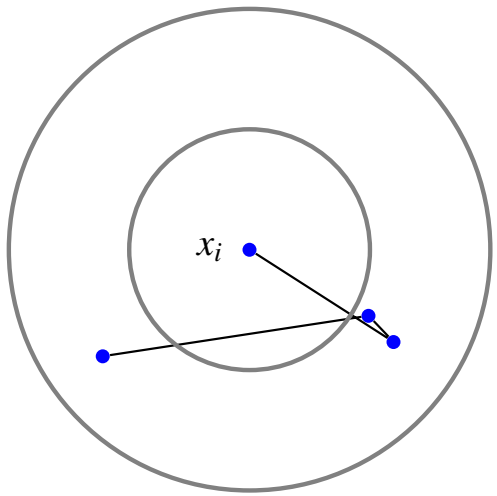
Example: A Markov chain on $\mathbb{R}^2$

We can define a very simple Markov chain by sampling

$$x_{i+1} \sim g(\,.\,|x_i, \sigma^2)$$

where $g(x|\mu, \sigma^2)$ is a spherical Gaussian with fixed variance. In other words, the transition distribution is

$$t(x_{i+1}|x_i) := g(x_{i+1}|x_i, \sigma^2)\ .$$



A Gaussian (gray contours) is placed around the current point x_i to sample x_{i+1}.

Recall: Finite case

- The invariant distribution P_{inv} is a distribution on the finite state space $\mathbf{X}$ of the MC (i.e. a vector of length $|\mathbf{X}|$).
- "Invariant" means that, if x_i is distributed according to P_{inv}, and we execute a step $x_{i+1} \sim t(\,.\,|x_i)$ of the chain, then x_{i+1} again has distribution P_{inv}.
- In terms of the transition matrix $\mathbf{T}$:

$$\mathbf{T} \cdot P_{\text{inv}} = P_{\text{inv}}$$

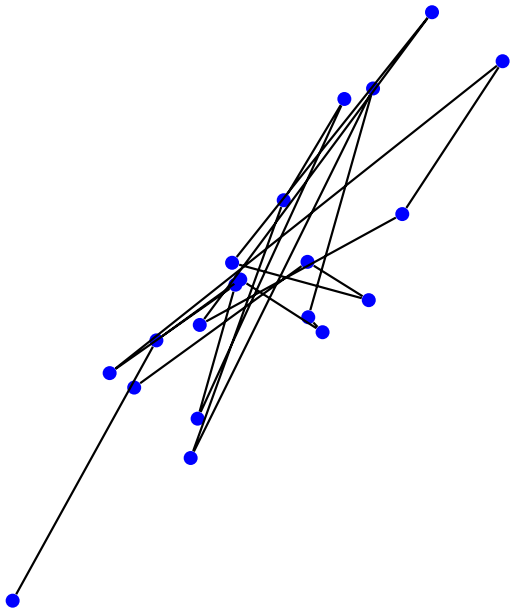
Continuous case

- $\mathbf{X}$ is now uncountable (e.g. $\mathbf{X} = \mathbb{R}^d$).
- The transition matrix $\mathbf{T}$ is substituted by the conditional probability t.
- A distribution P_{inv} with density p_{inv} is invariant if

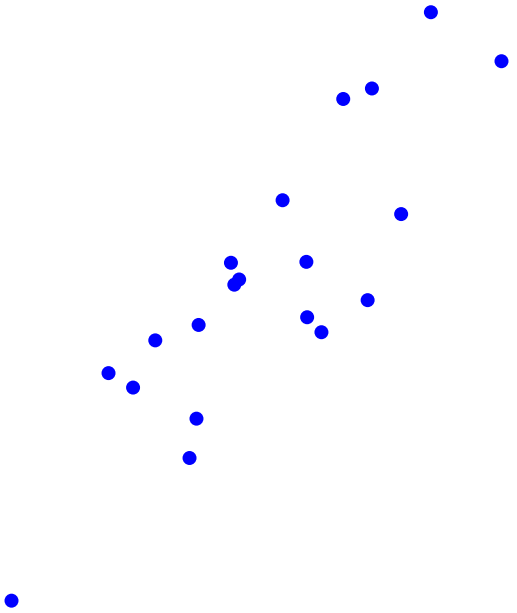
$$\int_{\mathbf{X}} t(y|x) p_{\text{inv}}(x) dx = p_{\text{inv}}(y)$$

This is simply the continuous analogue of the equation $\sum_i \mathbf{T}_{ij} (P_{\text{inv}})_i = (P_{\text{inv}})_j$.

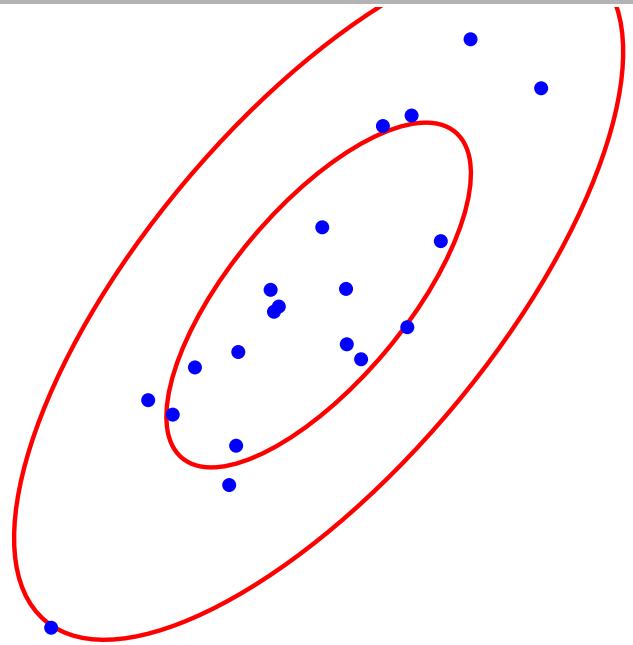
Markov Chain Sampling



We run the Markov chain n for steps. Each step moves from the current location x_i to a new x_{i+1}.



We "forget" the order and regard the locations $x_{1:n}$ as a random set of points.



If p (red contours) is both the invariant and initial distribution, each x_i is distributed as $x_i \sim p$.

Problems we need to solve

1. We have to construct a MC with invariant distribution p.
2. We cannot actually start sampling with $x_1 \sim p$; if we knew how to sample from p, all of this would be pointless.
3. Each point x_i is *marginally* distributed as $x_i \sim p$, but the points are *not* i.i.d.

Constructing the Markov Chain

Given is a continuous target distribution with density p.

Metropolis-Hastings (MH) kernel

1. We start by defining a conditional probability $r(y|x)$ on $\mathbf{X}$.
   r has nothing to do with p. We could e.g. choose $r(y|x) = g(y|x, \sigma^2)$, as in the previous example.
2. We define a **rejection kernel** A as

$$A(x_{n+1}|x_n) := \min\left\{1, \frac{r(x_i|x_{i+1})p(x_{i+1})}{r(x_{i+1}|x_i)p(x_i)}\right\}$$

   The normalization of p cancels in the quotient, so knowing $\tilde{p}$ is again enough.

3. We define the transition probability of the chain as

$$t(x_{i+1}|x_i) := r(x_{i+1}|x_i)A(x_{i+1}|x_i) + \delta_{x_i}(x_{i+1})c(x_i) \quad \text{where} \quad c(x_i) := \overbrace{\int r(y|x_i)(1-A(y|x_i))dy}^{\text{total probability that a proposal is sampled and then rejected}}$$

Sampling from the MH chain

At each step $i+1$, generate a proposal $x^* \sim r(\,.\,|x_i)$ and $U_i \sim \text{Uniform}[0,1]$.

- If $U_i \leq A(x^*|x_i)$, accept proposal: Set $x_{i+1} := x^*$.
- If $U_i > A(x^*|x_i)$, reject proposal: Set $x_{i+1} := x_i$.

Recall: Fundamental theorem on Markov chains

Suppose we sample $x_1 \sim P_{\text{init}}$ and $x_{i+1} \sim t(\,.\,|x_i)$. This defines a distribution P_i of x_i, which can change from step to step. If the MC is nice (recall: recurrent and aperiodic), then

$$P_i \to P_{\text{inv}} \qquad \text{for} \qquad i \to \infty\ .$$

Note: Making precise what aperiodic means in a continuous state space is a bit more technical than in the finite case, but the theorem still holds. We will not worry about the details here.

Implication

- If we can show that $P_{\text{inv}} \equiv p$, we do not have to know how to sample from p.
- Instead, we can start with *any* P_{init}, and will get arbitrarily close to p for sufficiently large i.

Burn-In and Mixing Time

The number m of steps required until $P_m \approx P_{\text{inv}} \equiv p$ is called the **mixing time** of the Markov chain. (In probability, there is a range of definitions for what exactly $P_m \approx P_{\text{inv}}$ means.)

In MC samplers, the first m samples are also called the **burn-in** phase. The first m samples of each run of the sampler are discarded:

$$\underbrace{x_1, \ldots, x_{m-1}}_{\text{Burn-in; discard.}}, \underbrace{x_m, x_{m+1}, \ldots}_{\text{Samples from (approximately) } p\text{; keep.}}$$

Convergence diagnostics

In practice, we do not know how large j is. There are a number of methods for assessing whether the sampler has mixed. Such heuristics are often referred to as **convergence diagnostics**.

Problem 2: Sequential Dependence

Even after burn-in, the samples from a MC are not i.i.d.

Strategy

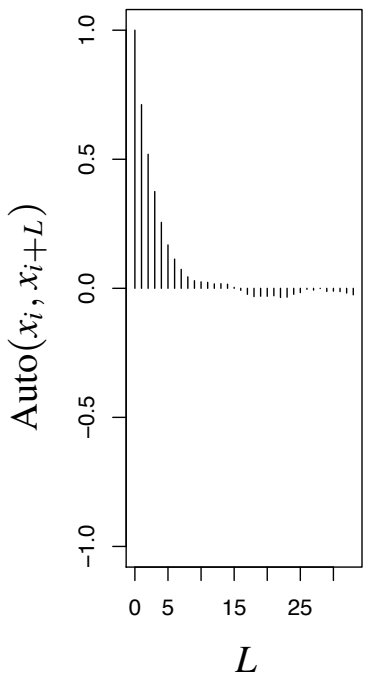
- Estimate empirically how many steps L are needed for x_i and x_{i+L} to be approximately independent. The number L is called the **lag**.
- After burn-in, keep only every Lth sample; discard samples in between.

Estimating the lag

The most commen method uses the **autocorrelation function**:

$$\text{Auto}(x_i, x_j) := \frac{\mathbb{E}[x_i - \mu_i] \cdot \mathbb{E}[x_j - \mu_j]}{\sigma_i \sigma_j}$$

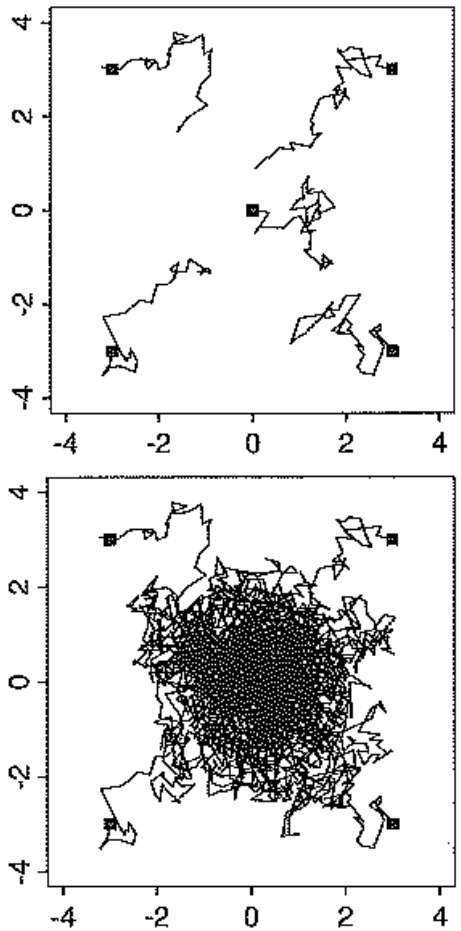
We compute $\text{Auto}(x_i, x_{i+L})$ empirically from the sample for different values of L, and find the smallest L for which the autocorrelation is close to zero.



There are about half a dozen popular convergence crieteria; the one below is an example.

Gelman-Rubin criterion

- Start several chains at random. For each chain k, sample x_i^k has a marginal distribution P_i^k.
- The distributions of P_i^k will differ between chains in early stages.
- Once the chains have converged, all $P_i = P_{\text{inv}}$ are identical.
- Criterion: Use a hypothesis test to compare P_i^k for different k (e.g. compare P_i^2 against null hypothesis P_i^1). Once the test does not reject anymore, assume that the chains are past burn-in.



Reference: A. Gelman and D. B. Rubin: "Inference from Iterative Simulation Using Multiple Sequences", *Statistical Science*, Vol. 7 (1992) 457-511.

STOCHASTIC HILL-CLIMBING

The Metropolis-Hastings rejection kernel was defined as:

$$A(x_{n+1}|x_n) = \min\left\{1, \frac{r(x_i|x_{i+1})p(x_{i+1})}{r(x_{i+1}|x_i)p(x_i)}\right\} .$$

Hence, we certainly accept if the second term is larger than 1, i.e. if

$$r(x_i|x_{i+1})p(x_{i+1}) > r(x_{i+1}|x_i)p(x_i) .$$

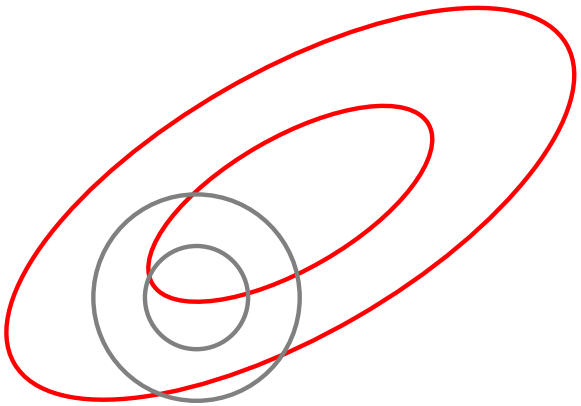
That means:

- We always accept the proposal x_{i+1} if it *increases* the probability under p.
- If it *decreases* the probability, we still accept with a probability which depends on the difference to the current probability.

Hill-climbing interpretation

- The MH sampler somewhat resembles a gradient ascent algorithm on p, which *tends* to move in the direction of increasing probability p.
- However:
  - The actual steps are chosen at random.
  - The sampler can move "downhill" with a certain probability.
  - When it reaches a local maximum, it does not get stuck there.

Everyone's favorite example: Two Gaussians



red = target distribution p
gray = proposal distribution r

- $\mathrm{Var}[r]$ too large:
  Will overstep p; many rejections.
- $\mathrm{Var}[r]$ too small:
  Many steps needed to achieve good coverage of domain.

If p is unimodal and can be roughly approximated by a Gaussian, $\mathrm{Var}[r]$ should be chosen as smallest covariance component of p.

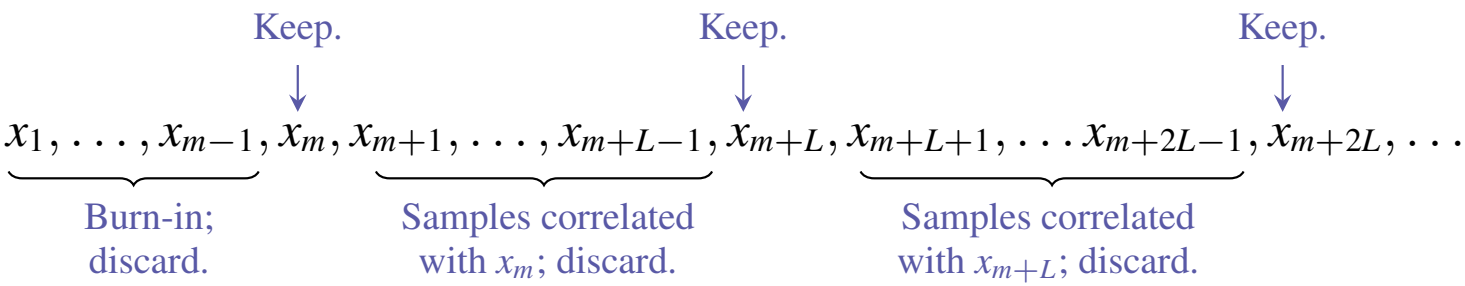
More generally

For complicated posteriors (recall: small regions of concentration, large low-probability regions in between) choosing r is much more difficult. To choose r with good performance, we already need to know something about the posterior.

There are many strategies, e.g. mixture proposals (with one component for large steps and one for small steps).

SUMMARY: MH SAMPLER

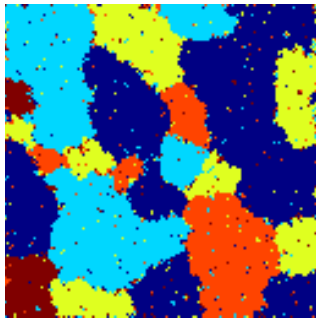
- MCMC samplers construct a MC with invariant distribution p.
- The MH kernel is one generic way to construct such a chain from p and a proposal distribution r.
- Formally, r does not depend on p (but arbitrary choice of r usually means bad performance).
- We have to discard an initial number m of samples as burn-in to obtain samples (approximately) distributed according to p.
- After burn-in, we keep only every Lth sample (where L = lag) to make sure the x_i are (approximately) independent.



Example: Burn-In Matters

This example is due to Erik Sudderth (Brown University).

MRFs as "segmentation" priors

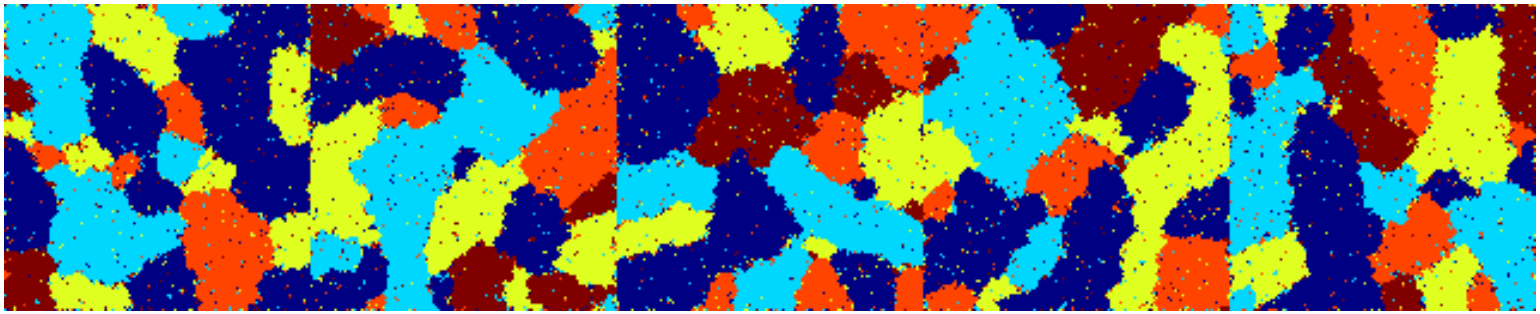


- MRFs where introduced as tools for image smoothing and segmentation by D. and S. Geman in 1984.
- They sampled from a Potts model with a Gibbs sampler, discarding 200 iterations as burn-in.
- Such a sample (after 200 steps) is shown above, for a Potts model in which each variable can take one out of 5 possible values.
- These patterns led computer vision researchers to conclude that MRFs are "natural" priors for image segmentation, since samples from the MRF resemble a segmented image.

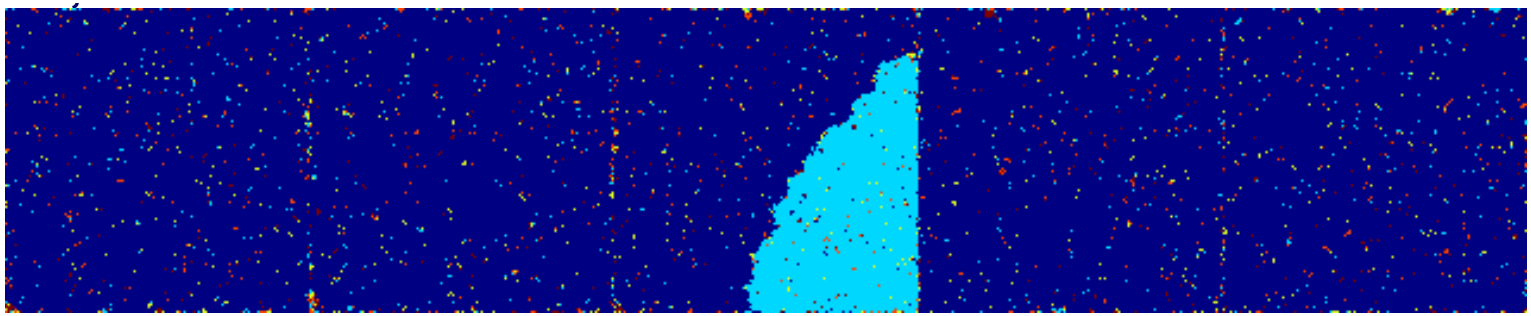
Example: Burn-In Matters

E. Sudderth ran a Gibbs sampler on the same model and sampled after 200 iterations (as the Geman brothers did), and again after 10000 iterations:

200 iterations



10000 iterations



Chain 1 Chain 5

- The "segmentation" patterns are not sampled from the MRF distribution $p \equiv P_{\text{inv}}$, but rather from $P_{200} \neq P_{\text{inv}}$.
- The patterns occur not because MRFs are "natural" priors for segmentations, but because *the sampler's Markov chain has not mixed*.
- MRFs are smoothness priors, not segmentation priors.

By far the most widely used MCMC algorithm is the Gibbs sampler.

Full conditionals

Suppose p is a distribution on $\mathbb{R}^{\mathrm{D}}$, so $x = (x_1, \ldots, x_{\mathrm{D}})$. The conditional probability of the entry x_i given all other entries,

$$p(x_d | x_1, \ldots, x_{d-1}, x_{d+1}, \ldots, x_{\mathrm{D}})$$

is called the **full conditional** distribution of x_{D}.

Gibbs sampling

The Gibbs sampler is a special case of the Metropolis-Hastings algorithm which uses the full conditionals to generate proposals.

- Gibbs sampling is only applicable if we can compute the full conditionals for each dimension i.
- If so, it provides us with a *generic* way to derive a proposal distribution.

Proposal distribution

Suppose p is a distribution on $\mathbb{R}^{\mathrm{D}}$, so each sample is of the form $x_i = (x_{i,1}, \ldots, x_{i,\mathrm{D}})$. We generate a proposal x_{i+1} coordinate by coordinate as follows:

$$
\begin{aligned}
x_{i+1,1} &\sim p(\,.\,|x_{i,2}, \ldots, x_{i,\mathrm{D}}) \\
&\vdots \\
x_{i+1,d} &\sim p(\,.\,|x_{i+1,1}, \ldots, x_{i+1,j-1}, x_{i,j+1}, \ldots, x_{i,d}) \\
&\vdots \\
x_{i+1,\mathrm{D}} &\sim p(\,.\,|x_{i+1,1}, \ldots, x_{i+1,\mathrm{D}-1})
\end{aligned}
$$

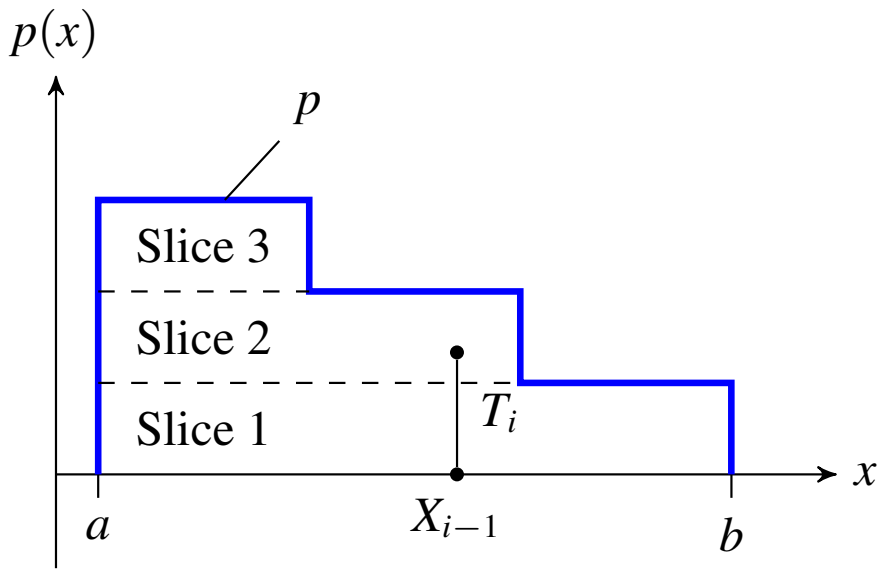
Note: Each new $x_{i+1,d}$ is immediately used in the update of the next dimension $d+1$.

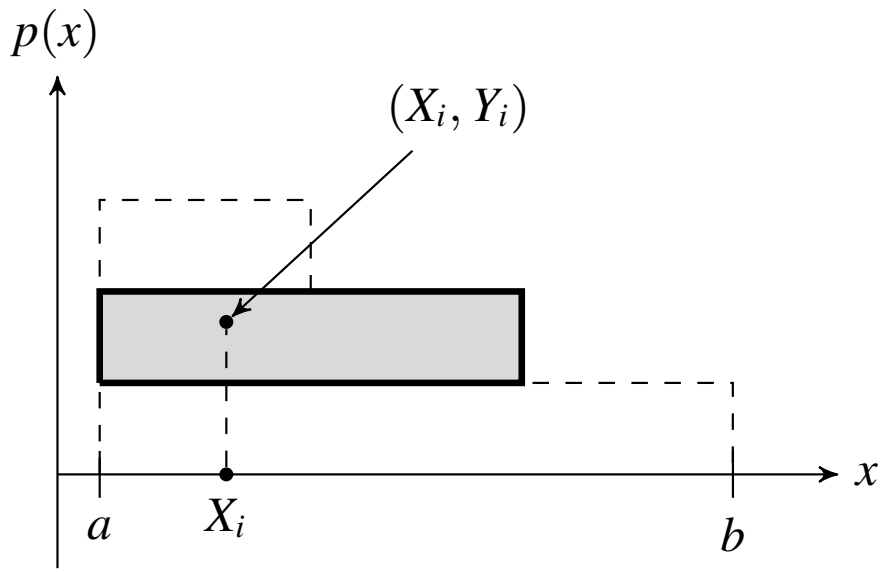
A Metropolis-Hastings algorithms with proposals generated as above is called a **Gibbs sampler**.

No rejections

It is straightforward to show that the Metropolis-Hastings acceptance probability for each $x_{i+1,d+1}$ is 1, so *proposals in Gibbs sampling are always accepted.*

SLICE SAMPLING





Start with any $x_1 \in [a, b]$. To generate X_i with $i > 1$:

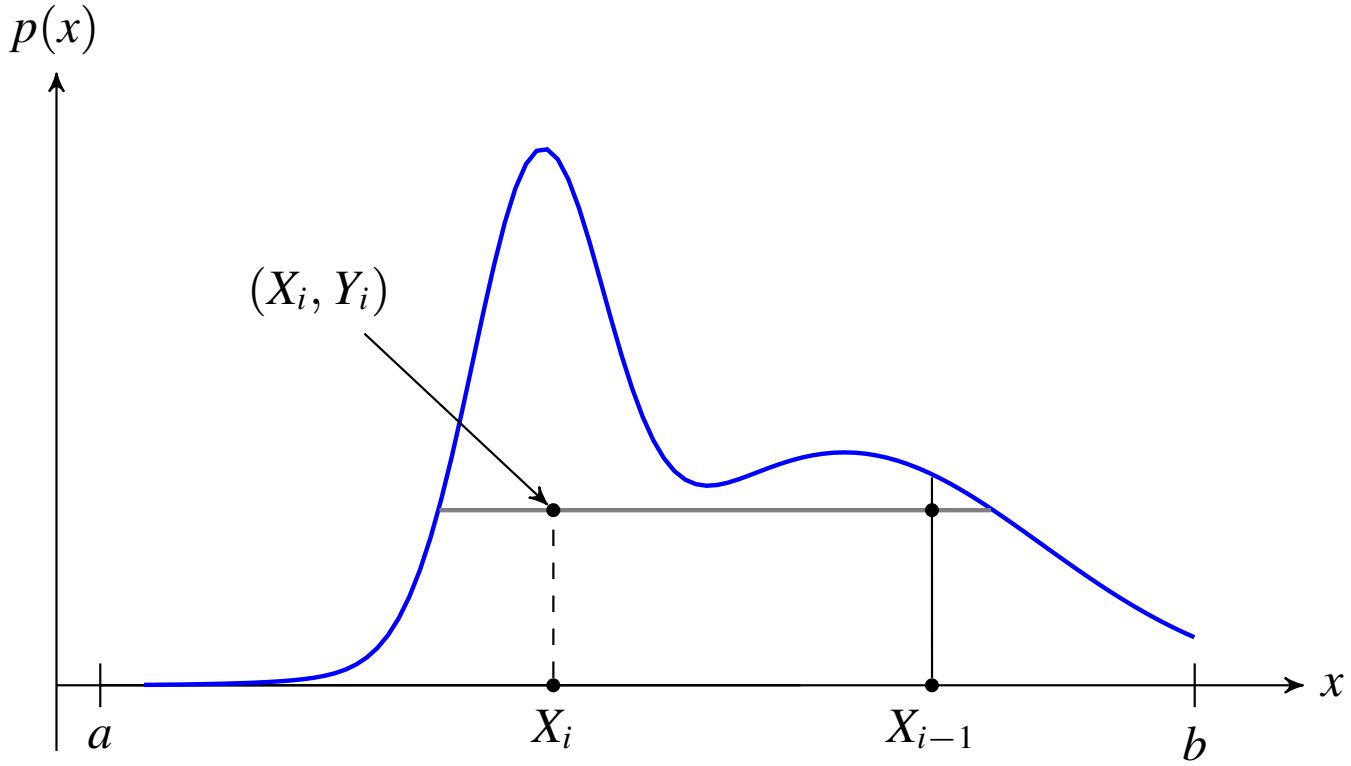
1. Choose one of the slices which overlap X_{i-1} uniformly at random:

   $T_i \sim \text{Uniform}[0, p(X_{i-1})]$ Select the slice k which contains (X_{i-1}, T_i) .

2. Regard slice k as a box and sample a point (X_i, Y_i) uniformly from this box.
3. Discard the vertical coordinate Y_i and keep X_i as the ith sample.

SLICE SAMPLING

Smooth density: Let slice thickness $\to 0$.



To generate x_i:

1. Choose one of the slices which overlap X_{i-1} uniformly at random:

$$Y_i \sim \text{Uniform}[0, p(X_{i-1})]$$

2. Sample X_i uniformly from gray line $= p^{-1}([Y_i, +\infty))$