Markov chain Monte Carlo (MCMC)

Kevin P. Murphy

Last updated November 3, 2006

* Denotes advanced topics that may be skipped on a first reading.

1 Monte Carlo integration

Suppose we want to evaluate the integral

$$I = \int_a^b h(x)dx \tag{1}$$

for some function h, where $x \in \mathcal{X}$, such as $\mathcal{X} = \mathbb{R}^D$. There are many numerical methods to do this (e.g., Simpson's rule), but they do not work well in high dimensions, due to the need to evaluate the function at a number of points which is exponential in D; this is called the **curse of dimensionality**.

An alternative approach for approximating I, which is notable for its simplicity, generality and scalability, is Monte Carlo integration. Let us start by writing

$$I = \int_a^b h(x)dx = \int_a^b w(x)p(x)dx \tag{2}$$

where $w(x) = h(x)(b-a)$ and $p(x) = 1/(b-a)$ is the pdf of a uniform random variable over (a, b). Hence

$$I = E_p[w(X)] \tag{3}$$

where $X \sim U(a, b)$. By sampling $X_{1:S} \sim U(a, b)$, by the **law of large nunbers** we have

$$\hat{I} = \frac{1}{S}\sum_{s=1}^{S} w(X_s) \approx I \tag{4}$$

The **standard error** of the estimate is

$$\hat{se} = \frac{s}{\sqrt{S}} \tag{5}$$

$$s^2 = \frac{1}{S-1}\sum_{s=1}^{S}(h(x_s) - \hat{I})^2 \tag{6}$$

So a $1-\alpha$ confidence interval of I is $\hat{I} \pm z_{\alpha/2}\hat{s}$, where z_q is the q'th quantile of a standard $\mathcal{N}(0, 1)$ variable.

For example, suppose $h(x) = x^3$. Then $I = \int_0^1 x^3 dx = 1/4$. We can approximate this integral using Monte Carlo sampling as follows.

```
% mcWasserman.m
% Demo of Monte Carlo integration from Wasserman p405
S = 10000;
xs = unifrnd(0,1,S,1);
samples = xs.^3;
Ihat = mean(samples)
se = sqrt(var(samples)/S)
```

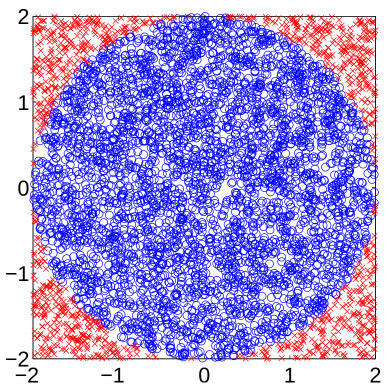


Figure 1: Estimating π by Monte Carlo integration. Blue circles are inside the circle, red crosses are outside. This figure was produced by `mcpi.m`.

We find $\hat{I} = 0.2525$ with standard error of 0.0028.

Let us consider another example. Suppose we want to estimate π. We know that the area of a circle with radius r is πr^2. The area of a circle is given by

$$I = \int_{-r}^{r} \int_{-r}^{r} I(x^2 + y^2 \leq r^2) dx dy \tag{7}$$

Hence $\pi = I/(r^2)$. Let us approximate this by Monte Carlo integration. Let $X, Y \sim U(-r, r)$, so using the above notation

$$\begin{aligned} w(x,y) &= (b_x - a_x)(b_y - a_y) I(x^2 + y^2 \leq r^2) && (8) \\ &= (2r)(2r) I(x^2 + y^2 \leq r^2) && (9) \\ &= 4r^2 I(x^2 + y^2 \leq r^2) && (10) \end{aligned}$$

We can implement this in Matlab as follows.

```
% mcpi.m
% Demo of monte carlo integration for estimating pi

r=2;
S=5000;
xs = unifrnd(-r,r,S,1);
ys = unifrnd(-r,r,S,1);
rs = xs.^2 + ys.^2;
inside = (rs <= r^2);
samples = 4*(r^2)*inside;
Ihat = mean(samples)
piHat = Ihat/(r^2)
se = sqrt(var(samples)/S)

figure(1);clf
outside = ~inside;
plot(xs(inside), ys(inside), 'bo');
hold on
plot(xs(outside), ys(outside), 'rx');
axis square
print(gcf,'-depsc','C:/kmurphy/figures/other/mc_pi.eps')
```

We find $\hat{\pi} = 3.1416$ with standard error 0.09. We can plot the points that are accepted/ rejected as in Figure 1.

We can generalize this to compute integrals of the form

$$I = \int h(x) p(x) dx \tag{11}$$

If $p(x)$ is uniform, we get the special case above. This is very useful in Bayesian inference (and in other applications). For example, if $h(x) = I(x_i = j)$, then $I = Eh(X) = p(X_i = j)$ is the marginal probability of X_i. We can approximate this as

$$\hat{I} = \frac{1}{S}\sum_{s=1}^{S} h(x^{(s)}) \tag{12}$$

The key question is how to draw $x^s \sim p(x)$. There are many methods e.g., rejection sampling, importance sampling, etc. The most popular method for high-dimensional problems is **Markov chain Monte Carlo (MCMC)**. (In a survey by *SIAM News*[1], MCMC was placed in the top 10 most important algorithms of the 20th century.)

2 Metropolis Hastings (MH) algorithm

In MCMC, we construct a Markov chain on $\mathcal{X}$ whose stationary distribution is the target density $\pi(x)$. (This is just some probability density function. In Bayesian inference, it would typically be a prior, $p(x)$, or a posterior, $p(x|y)$, but MCMC can be used in non Bayesian contexts, too.) By drawing dependent (correlated) states $X_0, X_1, X_2, \ldots,$ from the chain, we can perform Monte Carlo integration wrt π.

Let $x, x' \in \mathcal{X}$ be states in the chain. Let $q(x'|x)$ be an arbitrary, easy-to-sample from **proposal distribution**. Of course, such an arbitrary proposal cannot be expected to satisfy detailed balance, and hence may not be a stationary distribution. However, suppose (without loss of generality) that

$$q(x'|x)\pi(x) > q(x|x')\pi(x') \tag{13}$$

Then there is a factor $r(x'|x) \le 1$ such that the above inequality is balanced

$$q(x'|x)\pi(x)r(x'|x) = q(x|x')\pi(x') \tag{14}$$

Solving for r yields

$$r(x'|x) = \min\{1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}\} \tag{15}$$

This can be converted into an algorithm as follows.

1. Initialize X_0 arbitrarily.
2. For $s = 0, 2, \ldots$
   (a) Generate a proposed state $x' \sim q(x'|x_s)$
   (b) Evaluate the acceptance propability

$$\alpha = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} = \frac{\pi(x')/q(x'|x)}{\pi(x)/q(x|x')} \tag{16}$$

$$r(x'|x) = \min\{1, \alpha\} \tag{17}$$

   (c) Set

$$X_{s+1} = \begin{cases} x' & \text{with probability } r \\ x_s & \text{with probability } 1-r \end{cases} \tag{18}$$

For a given target distribution π, the proposal q is valid/ admissible if

$$\text{supp}(\pi) \subseteq \cup_x \text{supp}(q(\cdot|x)) \tag{19}$$

[1] `http://amath.colorado.edu/resources/archive/topten.pdf`,

where $\text{supp}(\pi) = \{x : \pi(x) > 0\}$ is the support of distribution π (i.e., the set of points with non-zero probability). This condition just says that our proposal must have a non-zero probability of moving to the states that have non-zero probability in the target. The resulting transition distribution of the algorithm is

$$p_M(x'|x) = q(x'|x)r(x'|x) + I(x' = x)(1 - \int q(x'|x)r(x'|x)dx') \tag{20}$$

An easy way to implement step (c) is to generate $U \sim U(0,1)$ and to set $X_{s+1} = x'$ if $U < r$, and to set $x_{s+1} = X_s$ otherwise.

Note that when evaluating α, we only need to know the target density p up to a normalization constant. In particular, suppose $\pi(x) = \frac{1}{Z}\pi'(x)$, where $\pi'(x)$ is an unnormalized distribution and Z is the normalization constant. Then

$$\alpha = \frac{(\pi'(x')/Z)\ q(x|x')}{(\pi'(x)/Z)\ q(x'|x)} \tag{21}$$

so the Z's cancel. Thus, using MH, we can sample from π even if we can only compute π'. Later we will see many examples where it is hard to evaluate Z but easy to evaluate π'.

If we have a **symmetric proposal distribution** $q(x'|x) = q(x|x')$, then the acceptance ratio simplifies to

$$\alpha \quad = \quad \frac{\pi(x')}{\pi(x)} \tag{22}$$

This is called the **Metropolis algorithm**. For example, it is common to use a Gaussian as a proposal distribution: $q(x'|x) = \mathcal{N}(x'|x, \sigma^2)$. This is called a **random walk MH** algorithm. It is crucial to pick the right σ^2 to ensure that a reasonable number (say 50%) of the proposals are accepted: see Figure 3.

In the Metropolis algorithm, if the new state x' is more probable than the current state x, the proposal is always accepted $r(x'|x) = 1$, otherwise it is accepted with probability $\pi(x')/\pi(x)$.

A special case of the Metropolis algorithm is when the proposal is independent of the current state: $q(x'|x) = q(x')$. Then the acceptance probability is

$$\alpha = \frac{\pi(x')/q(x')}{\pi(x)/q(x)} \tag{23}$$

This is called the **independence sampler**, and is similar to importance sampling.

Below is some generic MH code. If the proposal distribution is symmetric, it is not necessary to compute the actual probabilities $q(x'|x)$ and $q(x|x')$, it is only necessary to be able to sample from $q(x'|x)$.

```
function [samples, naccept] = MH(target, proposal, xinit, Nsamples, targetArgs, proposalArgs, proposalProb)
% Metropolis-Hastings algorithm
%
% Inputs
% target returns the unnormalized log posterior, called as 'p = exp(target(x, targetArgs{:}))'
% proposal is a fn, as 'xprime = proposal(x, proposalArgs{:})' where x is a 1xd vector
% xinit is a 1xd vector specifying the initial state
% Nsamples - total number of samples to draw
% targetArgs - cell array passed to target
% proposalArgs - cell array passed to proposal
% proposalProb  - optional fn, called as 'p = proposalProb(x,xprime, proposalArgs{:})',
%   computes q(xprime|x). Not needed for symmetric proposals (Metropolis algorithm)
%
% Outputs
% samples(s,:) is the s'th sample (of size d)
% naccept = number of accepted moves

if nargin < 5,  targetArgs = {}; end
if nargin < 6,  proposalArgs = {}; end
if nargin < 7, proposalProb = []; end

d = length(xinit);
samples = zeros(Nsamples, d);
x = xinit(:)';
naccept = 0;
logpOld = feval(target, x, targetArgs{:});
for t=1:Nsamples
```

```
  xprime = feval(proposal, x, proposalArgs{:});
  %alpha = feval(target, xprime, targetArgs{:})/feval(target, x, targetArgs{:});
  logpNew = feval(target, xprime, targetArgs{:});
  alpha = exp(logpNew - logpOld);
  if ~isempty(proposalProb)
    qnumer = feval(proposalProb, x, xprime, proposalArgs{:}); % q(x|x')
    qdenom = feval(proposalProb, xprime, x, proposalArgs{:}); % q(x'|x)
    alpha = alpha * (qnumer/qdenom);
  end
  r = min(1, alpha);
  u = rand(1,1);
  if u < r
    x = xprime;
    naccept = naccept + 1;
    logpOld = logpNew;
  end
  samples(t,:) = x;
end
```

2.1 Example: sampling from a mixture of two 1D Gaussians

We now show an example of how to call the above function where the target distribution is a mixture of two 1D Gaussians

$$\pi(x) = w_1 \mathcal{N}(x|\mu_1, \sigma_1) + w_2 \mathcal{N}(x|\mu_2, \sigma_2) \tag{24}$$

where $w_1 + w_2 = 1$ are called the **mixture weights**. The proposal is a 1D Gaussian $q(x'|x) = \mathcal{N}(x'|x, \sigma_p)$, where σ_p is a parameter of the proposal.

```
function mhDemoMOG()
% Demo of Metropolis-Hastings algorithm for sampling from
% a mixture of two 1D Gaussians using a Gaussian proposal.
% Based on code originally written by Nando de Freitas.

weights = [0.3 0.7];
mus = [0 10];
sigmas = [2 2];

Nsamples = 5000;
x = zeros(Nsamples,1);
sigma_prop = 10;                 % Standard deviation of the Gaussian proposal.

targetArgs = {weights, mus, sigmas};
proposalArgs = {sigma_prop};

seed = 1; randn('state', seed); rand('state', seed);
xinit = 20*rand(1,1); % initial state
[x, naccept] = MH(@target, @proposal, xinit, Nsamples,  targetArgs, proposalArgs);

% Let us check the asymmetric proposal works
%seed = 1; randn('state', seed); rand('state', seed);
%xinit = 20*rand(1,1); % initial state
%[x2, naccept] = MH(@target, @proposal, xinit, Nsamples,  targetArgs, proposalArgs, @proposalProb);
%assert(approxeq(x, x2))

% plot the histogram of samples
N_bins = 50;
Ns = [100 500 1000 Nsamples];
figure;
for i=1:4
  subplot(2,2,i)
  x_t = linspace(-10,20,1000);
  y_t = feval(@target, x_t, weights, mus, sigmas);
  [b,a] = hist(x(1:Ns(i)), N_bins);
  measure = a(2)-a(1); % bin width.
  area = sum(b*measure);
  bar(a,b/(area),'y')
  hold on;
  plot(x_t,y_t,'k','linewidth',2)
  axis([-10 20 0 .15])
  text(14,.1,sprintf('N=%d', Ns(i)))
end
```

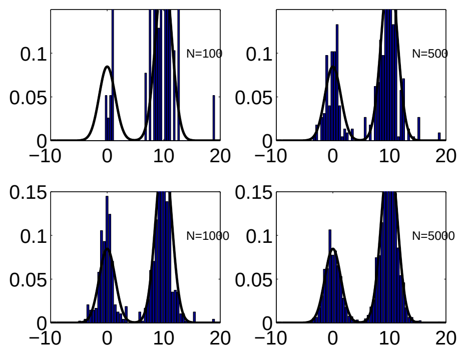


Figure 2: An example of the Metropolis Hastings algorithm for sampling from a mixture of two 1D Gaussians using a Gaussian proposal with variance $\sigma^2 = 10^2$. Figure produced using mhDemoMOG.m.

```
%%%%%%%%%%

function p = mogProb(x, mixWeights, mu, sigma)

% p(n) = sum_k w(k) N(x(n)|mu(k), sigma(k))
K = length(mixWeights);
N = length(x);
p = zeros(N,1);
for k=1:K
  p = p + mixWeights(k)*mvnpdf(x(:), mu(k), sigma(k));
end

function  p = target(x, mixWeights, mus, sigmas)
p = log(mogProb(x, mixWeights, mus, sigmas));

function xp = proposal(x, sigma_prop)
xp = x + sigma_prop*randn(1,1);

function p = proposalProb(x, xprime, sigma_prop)
p = normpdf(x, xprime, sigma_prop);
```

Some typical results are shown in Figure 2. It is important to set the variance of the proposal correctly: see Figure 3.

2.2 Example: sampling from a 2D Gaussian

As another example, below we show code to sample from a 2D Gaussian. We use the proposal $q(x'|x) = \mathcal{N}(x'|x, \sigma * I_2)$, where I_2 is the 2×2 identity matrix. We consider $\sigma = 0.01$, which does not mix well, and $\sigma = 1$, which does mix well: see Figure 9.

```
function mhDemoGauss2d()
% Demo of Metropolis-Hastings algorithm for sampling from
% a 2D Gaussian using a Gaussian proposal.
% Compare to gibbsGaussDemo.m

Nsamples = 5000;
burnin  = 1000;

%sigma = 0.01; % does't mix
sigma = 1; % mixes
SigmaProp = sigma*eye(2);

mu = [0 0];
C = [2 1; 1 1];

targetArgs = {mu, C};
proposalArgs = {SigmaProp};
```

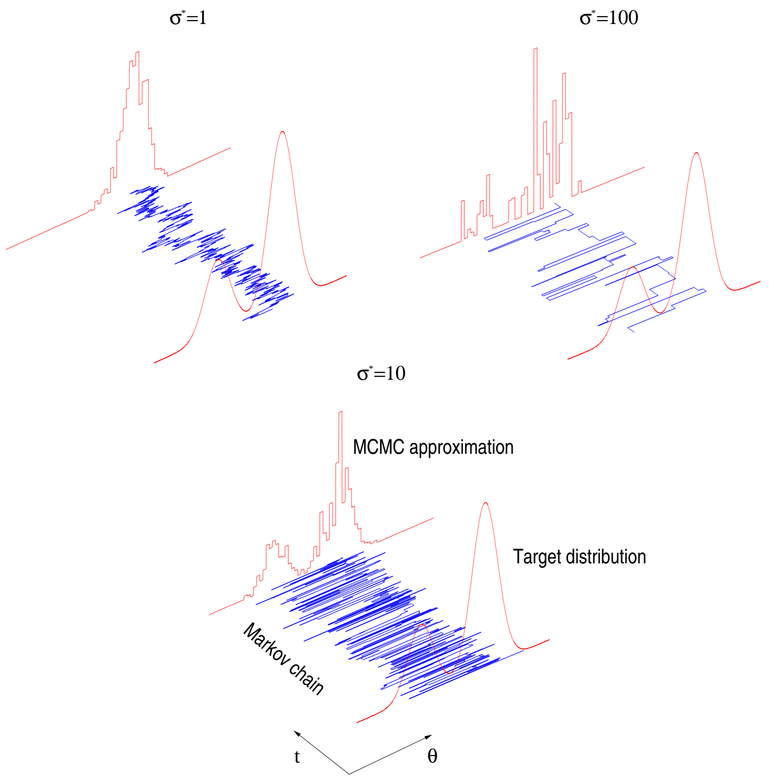


Figure 3: An example of the Metropolis Hastings algorithm for sampling from a mixture of two 1D Gaussians using a Gaussian proposal with different variances. Source: [AdFDJ03].

```
% try different starting seeds to check if mixing
seeds = [1 2 3];
figure; colors = {'r', 'g', 'b', 'k'};
samples = zeros(Nsamples-burnin, 2, length(seeds));
for c=1:length(seeds)
  seed = seeds(c);
  randn('state', seed); rand('state', seed);
  xinit = 20*rand(2,1); % initial state
  [tmp, naccept] = MH(@target, @proposal, xinit, Nsamples,  targetArgs, proposalArgs);
  samples(:,:,c) = tmp(burnin+1:end,:);
  plot(samples(:,1,c), colors{c});
  hold on
end
Rhat1 = EPSR(squeeze(samples(:,1,:)))
Rhat2 = EPSR(squeeze(samples(:,2,:)))
title(sprintf('sigmaProposal␣=␣%3.2f,␣Rhat=%5.3f', sigma, Rhat1))


figure;
h=draw_ellipse(mu', C);
set(h, 'linewidth', 3, 'color', 'r');
axis equal
set(gca, 'xlim', [-5 5]);
set(gca, 'ylim', [-5 5]);
hold on
ndx = 1:10:size(samples,1); % only plot subset of points
plot(samples(ndx,1), samples(ndx,2), 'k.');

% Plot 1D exact and approximate marginals
for i=1:2
  figure;
  Nbins = 100;
  [h, xs] = hist(samples(:,1),Nbins);
  binWidth = xs(2)-xs(1);
  bar(xs, normalise(h)/binWidth);
  hold on
  ps = normpdf(xs, mu(i), sqrt(C(i,i)));
  plot(xs, ps, '-');
  title(sprintf('x%d', i))
end

%%%%%%%%%%

function  p = target(x, mu, Sigma)
p = log(mvnpdf(x(:)', mu, Sigma));

function xp = proposal(x, SigmaProp)
xp = mvnrnd(x, SigmaProp);
```

2.3 Example: Binomial distribution with non conjugate prior

Suppose we make a new version of a product and ask $N = 20$ people if they prefer it to the old version; $X = 12$ people say yes. Let θ be probability they prefer the new version. We want to compute $\pi(\theta) = p(\theta|X = 12, N = 20)$.[2]

Let $X \sim Bino(N, \theta)$. Suppose the prior is flat but we know that at least half the people will prefer the new version, which we encode as $\theta \sim U(0.5, 1)$. In other words, our probability model is

$$p(X|\theta, N) = \binom{N}{X} \theta^X (1-\theta)^{N-X} \quad (25)$$

$$p(\theta) = \frac{1}{1-0.5} I(0.5 \leq \theta \leq 1) \quad (26)$$

$$\pi(\theta) = p(\theta|X, N) \propto \theta^X (1-\theta)^{N-X} I(0.5 \leq \theta \leq 1) \quad (27)$$

Note that the truncated uniform prior is not conjugate to the binomial likelihood. We can compute the posterior using MH. Although it is possible to use proposals that only propose valid values of $\theta \in [0.5, 1]$, it is common to transform

[2]This example is from Brani Vidakovic.

such constrained parameters to unconstrained form. Define

$$\phi = \log \frac{\theta - 0.5}{1 - \theta} \tag{28}$$

so $\phi \in (-\infty, \infty)$, with inverse transform

$$\theta = \frac{0.5 + e^{\phi}}{1 + e^{\phi}} \tag{29}$$

Now we can use a Gaussian proposal on ϕ. However, we have to compute the transformed target density. By the **change of variables formula**, we have

$$\begin{aligned} p(\theta) &= p(\phi) |\frac{d\phi}{d\theta}| && (30) \\ &= I(0.5 \leq \theta \leq 1) J && (31) \\ &= I(-\infty \leq \phi \leq \infty) J && (32) \end{aligned}$$

where the **Jacobian** is given by

$$\begin{aligned} J &= |\frac{d\phi}{d\theta}| && (33) \\ &= \frac{0.5 e^{\phi}}{(1 + e^{\phi})^2} && (34) \end{aligned}$$

Since

$$1 - \theta = \frac{1 + e^{\phi} - 0.5 - e^{\phi}}{1 + e^{\phi}} = \frac{0.5}{1 + e^{\phi}} \tag{35}$$

the posterior of the transformed variable is

$$\begin{aligned} p(\phi|X) &\propto p(X|\phi) p(\phi) && (36) \\ &\propto \left(\frac{0.5 + e^{\phi}}{1 + e^{\phi}} \right)^X \left(\frac{0.5}{1 + e^{\phi}} \right)^{N-X} \frac{0.5 e^{\phi}}{(1 + e^{\phi})^2} && (37) \\ &\propto \frac{(0.5 + e^{\phi})^X e^{\phi}}{(1 + e^{\phi})^{N+2}} && (38) \\ &= \frac{(0.5 + e^{\phi})^{12} e^{\phi}}{(1 + e^{\phi})^{22}} && (39) \end{aligned}$$

We can use this is a target density for MH, and then transform the ϕ samples back to the θ represesentation using Equation 29. See Figure 4 for some results using a Gaussian proposal with $\sigma = 0.5$, and Figure 5 for some results using a Gaussian proposal with $\sigma = 10$, which does not mix as well.

3 Why MH works

Recall from the Markov chain chapter that a chain satisfies **detailed balance** if

$$T_{jk} \pi_j = T_{kj} \pi_k \tag{40}$$

We also showed that if a chain satisfies detailed balance, then π is its stationary distribution.

In MCMC, we often deal with continuous state spaces, so we will write $p(x'|x)$ for the transition probability from x to x', instead of T_{ij}, and $\pi(x)$ for the stationary distribution, instead of π_k. In this case, detailed balance means

$$\pi(x) p(x'|x) = \pi(x') p(x'|x) \tag{41}$$

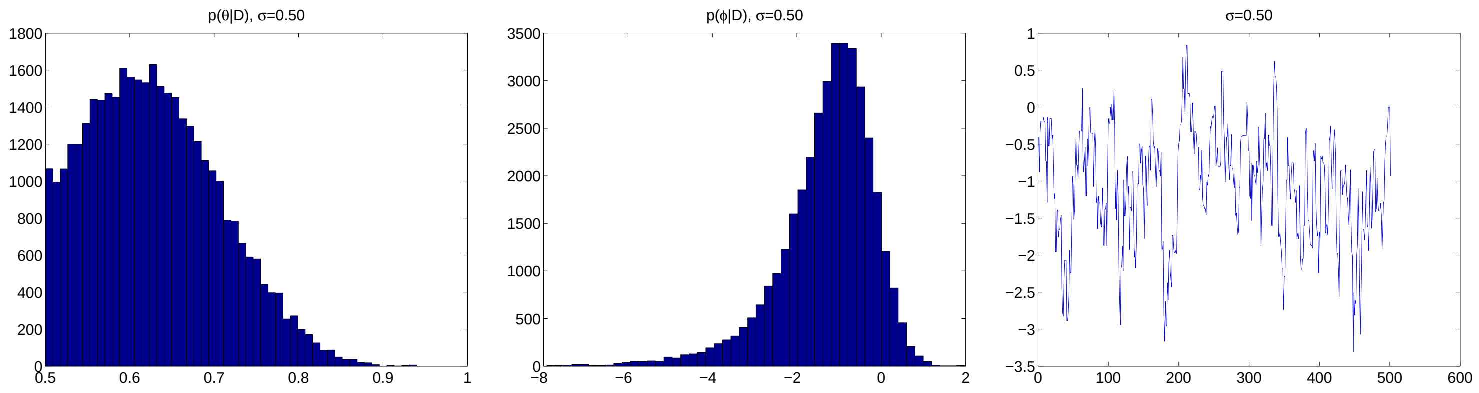


Figure 4: An example of the Metropolis algorithm for sampling from a binomial distribution with uniform prior using a Gaussian proposal with $\sigma = 0.5$. We used 40,000 samples and a burnin of 2000. Left: samples of the original parameter θ. The peak is near the MLE of $\hat{\theta}^{ML} = 0.6$. Middle: samples of the transformed parameter ϕ. Right: plot of the last 500 samples of ϕ. Figure produced using `mhDemoBino.m` (exercise).

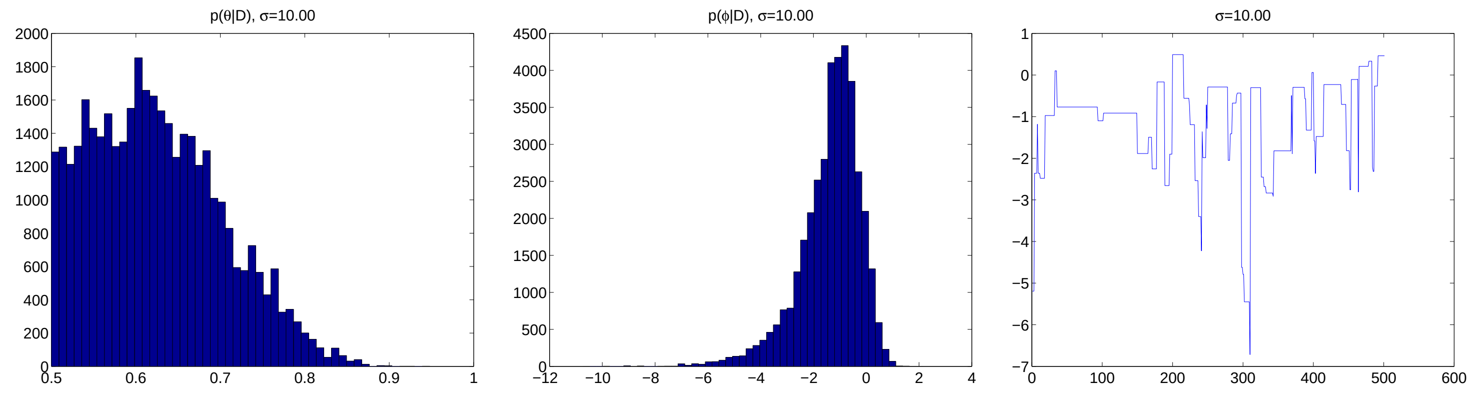


Figure 5: Same as Figure 4, except the Gaussian proposal has $\sigma = 10$. On the right we see the chain is not mixing is well, so the histograms are narrower and more blocky.

This implies π is a stationary distribution, since

$$\int \pi(x')p(x|x')dx' = \int \pi(x)p(x'|x)dx' = \pi(x)\int p(x'|x)dx' = \pi(x) \tag{42}$$

Our goal is to show that the MH algorithm defines a transition function that satisfies detailed balance and hence that p is its stationary distribution.

Consider two states x and x'. Either

$$\pi(x)q(x'|x) < \pi(x')q(x|x') \tag{43}$$

or

$$\pi(x)q(x'|x) > \pi(x')q(x|x') \tag{44}$$

We will ignore ties (which occur with probability zero for continuous distributions). Without loss of generality, assume that $\pi(x)q(x'|x) > \pi(x')q(x|x')$.

$$\alpha(x'|x) = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} < 1 \tag{45}$$

Hence from Equation 17, we have $r(x'|x) = \alpha(x'|x)$, but $r(x|x') = 1$ since $\alpha(x|x') > 1$.

Now to move from x to x' we must first propose x' and then accept it. Hence

$$p(x'|x) = q(x'|x)r(x'|x) = q(x'|x)\frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} = \frac{\pi(x')}{\pi(x)}q(x|x') \tag{46}$$

Hence

$$\pi(x)p(x'|x) = \pi(x')q(x|x') \tag{47}$$

The backwards probability is

$$p(x|x') = q(x|x')r(x|x') = q(x|x') \tag{48}$$

since $r(x|x') = 1$. Inserting this into Equation 47 we get

$$\pi(x)p(x'|x) = \pi(x')p(x|x') \tag{49}$$

so detailed balance holds.

4 Simulated annealing

Simulate annealing (SA) is an optimization algorithm, i.e., it attempts to find a global optimum

$$x^* \in \arg\max_x \pi(x) \tag{50}$$

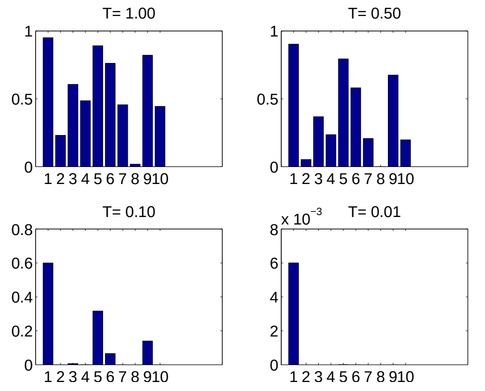
This can be implemented by modifying the MH algorithm to use a target distribution that is “cooled” over time: at iteration s, we use $\pi_s(x) = \pi(x)^{1/T_s}$ instead of $p(x)$ as the target, where T_s is the temperature at step s. It is common to use **exponential cooling**:

$$T_s = T_0 C^s \tag{51}$$

where T_0 is the initial temperature (often $T_0 = 1$) and C is the cooling rate (often $C = 0.9$). (These particular values are heuristically chosen.)

At high temperature, $\pi_s(x)$ will be almost uniform, allowing the algorithm to move freely between all states. As the temperature drops, “bumps” in the probability distribution start to appear. As $T \rightarrow 0$, only the largest peak survives: see Figure 6 for an example. If the temperature is cooled slowly enough, one can prove this algorithm will find a global optimum, essentially by “tracking” the peaks. However, “slowly enough” in practice might take exponential time. Designing good **annealing schedules** is a difficult problem. **Simulated tempering** is a related MCMC method in which the temperature is a stochastic variable and does not decrease deterministically.

Note that there are many other methods for finding the optima of non differentiable functions, such as **hill climbing**, **stochastic local search**, **tabu search**, **genetic algorithms**, etc. The unique thing about SA is its theoretical guarantee. However, in practice, some of the above heuristic methods can work much better. (For functions that are differentiable, a variety of numerical methods, such as **Newton’s method**, can be used. We will encounter these later.)



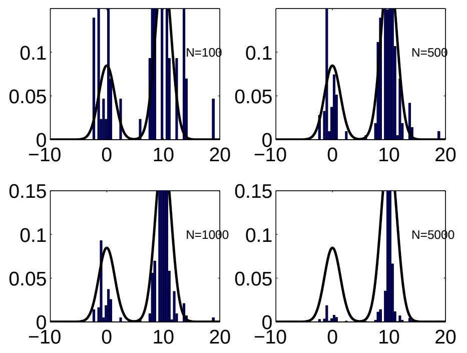


Figure 6: An example of simulated annealing. Left: we create a random distribution on 10 states, $\pi'(x)$, and plot $\pi'(x)^{1/T}$ for $T = 1, 0.5, 0.1, 0.01$. We see that the smallest peaks die off exponentially faster than the largest peaks. This figure was produced using `SAdemoHisto.m`. Right: we apply this cooling idea to a mixture of two 1D Gaussians. We use the cooling schedule $T_s = 0.995^{s-1}$, starting at $T_1 = 1$. We plot samples drawn from this distribution for $s = 100, 500, 1000, 5000$. At the end, most of the samples come from the peak. This figure was produced using `SAdemoMOG.m`.

5 Gibbs sampling

Gibbs sampling is a way to sample from a joint distribution one variable at a time. In particular, we use a sequence of proposals. To generate sample x^{s+1}, we sample each component in turn:

1. $x_1^{s+1} \sim p(x_1|x_2^s, \ldots, x_D^s)$
2. $x_2^{s+1} \sim p(x_2|x_1^{s+1}, x_3^s, \ldots, x_D^s)$
3. $x_i^{s+1} \sim p(x_i|x_{1:i-1}^{s+1}, x_{i+1:D}^s)$
4. $x_D^{s+1} \sim p(x_D|x_1^{s+1}, \ldots, x_{D-1}^{s+1})$

where $p(x_i|\cdot) = \pi(x_i|\cdot)$ are conditionals of the target distribution π. See Figure 7 for an example. Note that we can update the components in any order we wish.

We now prove that the acceptance rate of this proposal is 1. Let x_{-i} be all the variables except i. Let the proposal be denoted by

$$q((x_i', x_{-i})|(x_i, x_{-i})) = p(x_i'|x_{-i}) \tag{52}$$

Then

$$\begin{aligned} \alpha &= \frac{p(x')q(x|x')}{p(x)q(x'|x)} && (53) \\ &= \frac{p(x_i'|x_{-i})p(x_{-i})p(x_i|x_{-i})}{p(x_i|x_{-i})p(x_{-i})p(x_i'|x_{-i})} && (54) \\ &= 1 && (55) \end{aligned}$$

Gibbs sampling is very popular because it is easy to use, since there is no need to design a proposal distribution. All it requires is that the **full conditionals** $p(x_i|x_{-i})$ be easy to sample from, which is often the case, especially in **hierarchical Bayesian models**, as we will see later.

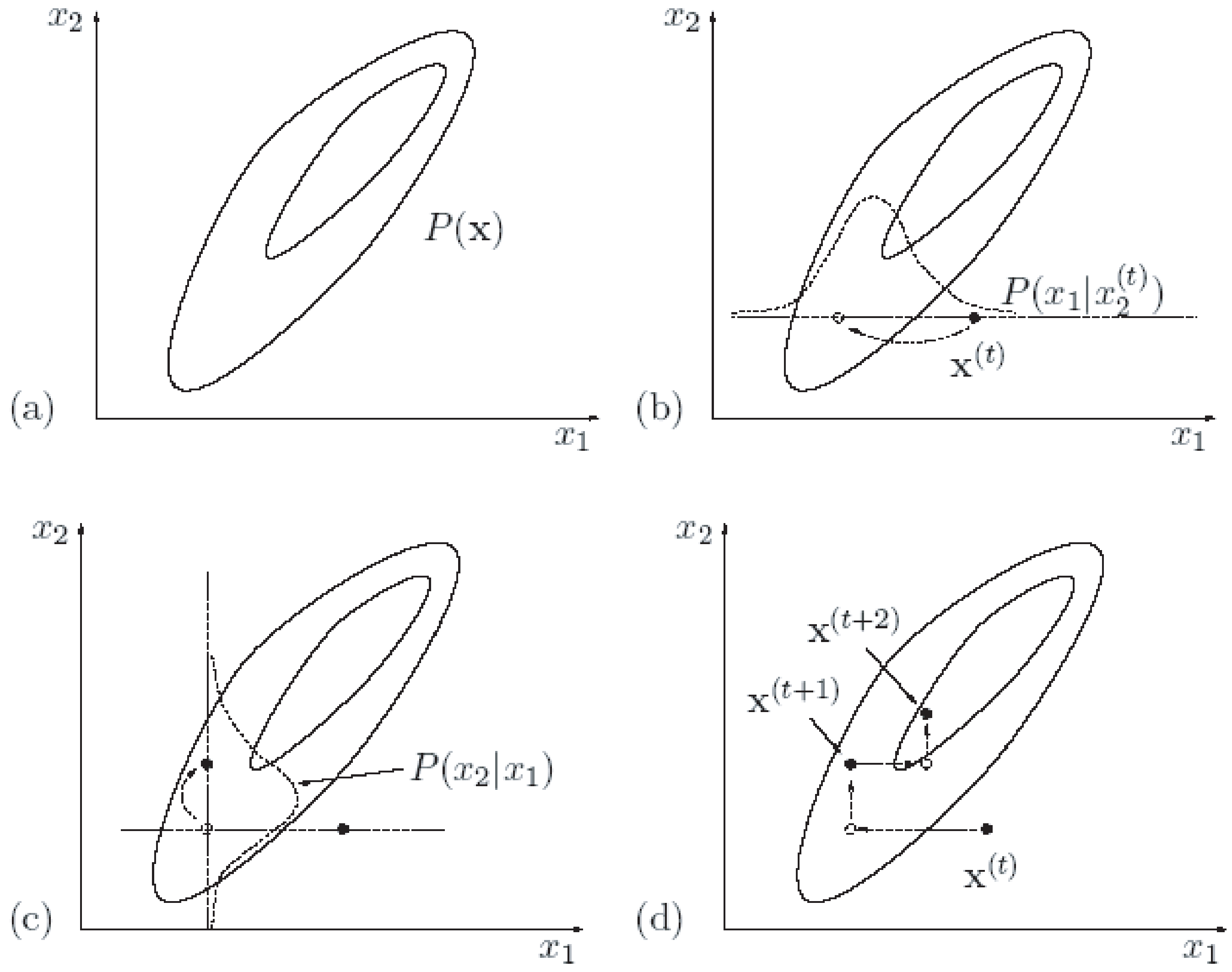


Figure 7: Example of Gibbs sampling in a 2D Gaussian. Source: [Mac03].

5.1 Example: Gibbs sampling for multivariate Gaussians

As a demonstration of Gibbs sampling, let us try sampling from a multivariate Gaussian

$$\mathcal{N}(\vec{x}|\vec{\mu}, \Sigma) \overset{\text{def}}{=} \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp[-\tfrac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})] \tag{56}$$

Although it is possible to sample directly from a Gaussian, we shall use Gibbs sampling for didactic reasons.

We need the following key fact. If we partition a Gaussian random vector $x_{1:D} = (x_1, x_2)$ and its parameteres

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{57}$$

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \tag{58}$$

then we can compute $p(x_1|x_2)$ as follows:

$$p(x_1|x_2) = \mathcal{N}(x_1; \mu_{1|2}, \Sigma_{1|2}) \tag{59}$$

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \tag{60}$$

$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{61}$$

The following function computes $p(X_A|X_B = x) = \mathcal{N}(\mu_{A|B}, \Sigma_{A|B})$:

```
function [muAgivenB, sigmaAgivenB] = gaussCondition(mu, Sigma, a, x)

D = length(mu);
b = setdiff(1:D, a);
muA = mu(a); muB = mu(b);
SAA = Sigma(a,a);
SAB = Sigma(a,b);
SBB = Sigma(b,b);
SBBinv = inv(SBB);
muAgivenB = mu(a) + SAB*SBBinv*(x(b)-mu(b));
sigmaAgivenB = SAA - SAB*SBBinv*SAB';
```

Using this, we can implement Gibbs sampling as follows.

```
function samples = gibbsGauss(mu, Sigma, xinit, Nsamples)
% Gibbs sampling for a multivariate Gaussian
%
% Input:
% mu(1:D) is the mean
% Sigma(1:D, 1:D) is the covariance
% xinit(1:D) is the initial state
% Nsamples = number of samples to draw
%
% Output:
% samples(t,:)

D = length(mu);
samples = zeros(Nsamples, D);
x = xinit(:)';
for s=1:Nsamples
  for i=1:D
    [muAgivenB, sigmaAGivenB] = gaussCondition(mu, Sigma, i, x);
    x(i) = normrnd(muAgivenB, sqrt(sigmaAGivenB));
  end
  samples(s,:) = x;
end
```

A demo of using this code is shown below, which results in Figure 8.

```
% gibbsGaussDemo
% Use Gibbs sampling to sample from a 2D Gaussian

S = 5000;
mu = [1 1];
```

```
C = [2 1; 1 1];

% try different starting seeds to check if mixing
seeds = [1 2 3];
figure; colors = {'r', 'g', 'b', 'k'};
for seedi=1:length(seeds)
  seed = seeds(seedi);
  rand('state', seed); randn('state', seed);
  xinit = 20*rand(2,1); % initial state
  samples = gibbsGauss(mu, C, xinit, S);
  burnin  = 1000;
  samples = samples(burnin+1:end,:);
  plot(samples(:,1), colors{seedi});
  hold on
end

figure;
h=draw_ellipse(mu', C);
set(h, 'linewidth', 3, 'color', 'r');
axis equal
set(gca, 'xlim', [-5 5]);
set(gca, 'ylim', [-5 5]);
hold on
ndx = 1:10:size(samples,1); % only plot subset of points
plot(samples(ndx,1), samples(ndx,2), 'k.');

% Plot 1D exact and approximate marginals
for i=1:2
  figure;
  Nbins = 100;
  [h, xs] = hist(samples(:,1),Nbins);
  binWidth = xs(2)-xs(1);
  bar(xs, normalise(h)/binWidth);
  hold on
  ps = normpdf(xs, mu(i), sqrt(C(i,i)));
  plot(xs, ps, '-');
  title(sprintf('x%d', i))
end
```

This demo uses the following handy function.

```
function h = draw_ellipse(x, c, outline_color, fill_color)
% DRAW_ELLIPSE(x, c, outline_color, fill_color)
%   Draws ellipses at centers x with covariance matrix c.
%   x is a matrix of columns.  c is a positive definite matrix.
%   outline_color and fill_color are optional.
% Written by Tom Minka

n = 40;                                 % resolution
radians = [0:(2*pi)/(n-1):2*pi];
unitC = [sin(radians); cos(radians)];
r = chol(c)';

if nargin < 3
  outline_color = 'g';
end

h = [];
for i=1:size(x,2)
  y = r*unitC + repmat(x(:, i), 1, n);
  if nargin < 4
    h = [h line(y(1,:), y(2,:), 'Color', outline_color)];
  else
    h = [h fill(y(1,:), y(2,:), fill_color, 'EdgeColor', outline_color)];
  end
end
```

5.2 Metropolis within Gibbs *

If we cannot easily sample from the full conditionals, we can use the MH algorithm inside the Gibbs algorithm. Specifically, to sample from

$$x_i^{s+1} \sim p(x_i | x_{1:i-1}^{s+1}, x_{i+1:D}^{s}) \tag{62}$$

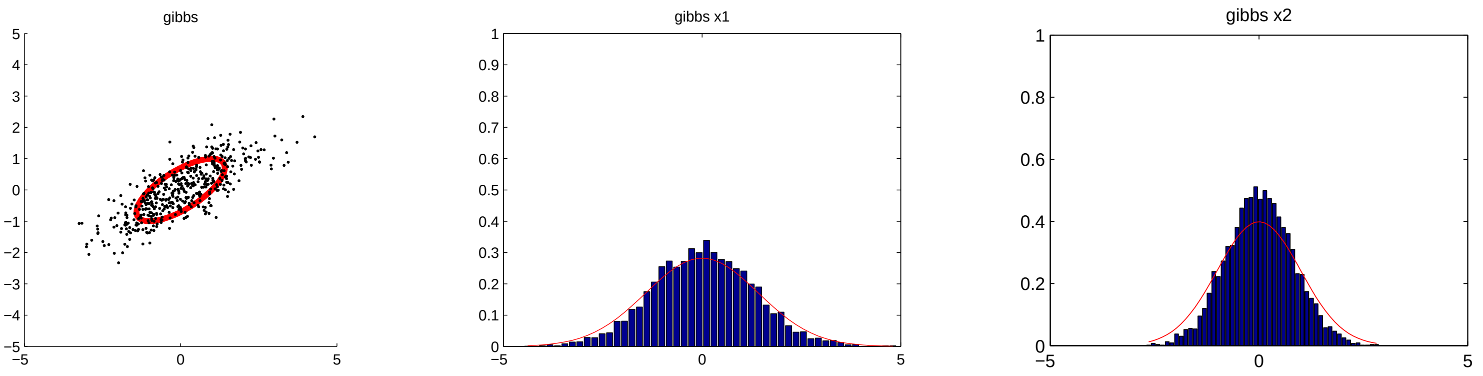


Figure 8: Example of Gibbs sampling on a 2D Gaussian. This figure was produced by `gibbsGaussDemo.m`.

we proceed in 3 steps:

1. Propose $x'_i \sim q(x'_i|x^s_i)$
2. Compute the acceptance probability $r_i = \min(1, \alpha_i)$ where

$$\alpha_i = \frac{p(x^{s+1}_{1:i-1}, x'_i, x^s_{i+1:D})/q(x'|x^s_i)}{p(x^{s+1}_{1:i-1}, x^s_i, x^s_{i+1:D})/q(x^s_i|x'_i)} \tag{63}$$

3. Sample $u \sim U(0,1)$ and set $x^{s+1}_i = x'_i$ if $u < r$, and set $x^{s+1}_i = x^s_i$ otherwise.

6 Convergence

We start MCMC from an arbitrary initial state. The amount of time it takes to **converge to its stationary distribution** is called the **mixing time** or **burnin time**. Once the chain has mixed, it is "safe" to start collecting samples. Since the samples are correlated, it is common to pick a subset of them (say every 10'th), a practice known as **thinning**. This reduces the storage requirements but does not improve the computational or statistical efficiency.

How do we know the chain has converged? This is a hard theoretical question. There are many heuristics, but none are guaranteed to work. In particular, some diagnostics may falsely claim the chain has converged yielding incorrect results. A common approach is to run multiple chains from very different **overdispersed** starting points, and to plot the samples of some variables of interest. If the chain has mixed, it should have "forgotten" where it started from. Figure 9 shows some examples of chains that have mixed and not mixed.

6.1 EPSR *

We can assess convergence more quantitatively as follows. Suppose we collect S samples from each of C chains, $x_{isc}, i = 1:D, s = 1:S, c = 1:C$. Let y be a scalar quantity of interest computed from $x_{1:D}$ (e.g., one of the variables, say $y = x_i$). Define the within-sequence mean and overall mean as

$$\overline{y}_{\cdot c} = \frac{1}{S}\sum_{s=1}^{S} y_{sc} \tag{64}$$

$$\overline{y}_{\cdot\cdot} = \frac{1}{C}\sum_{c=1}^{C} \overline{y}_{\cdot c} \tag{65}$$

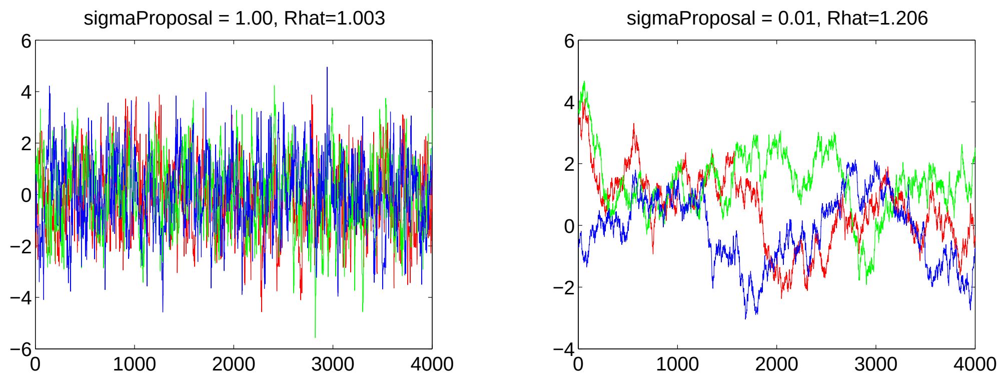


Figure 9: Left: Example of 3 chains that have mixed. Right: Example of 3 chains that have not mixed. This figure was generated by `mhDemoGauss2d.m`.

Define the between sequence and within sequence variance as

$$B = \frac{S}{C-1}\sum_{c=1}^{C}(\overline{y}_{\cdot c} - \overline{y}_{\cdot\cdot})^2 \quad (66)$$

$$W = \frac{1}{C}\sum_{c=1}^{C}\left[\frac{1}{S-1}\sum_{s=1}^{S}(y_{sc} - \overline{y}_{\cdot c})^2\right] \quad (67)$$

We can now construct two estimates of the variance of y. The first estimate is W: this should underestimate Var (y) if the chains have not ranged over the full posterior. The second estimate

$$\hat{V} = \frac{S-1}{S}W + \frac{1}{S}B \quad (68)$$

is an estimate of Var (y) that is unbiased under stationarity but is an overestimate the the starting points were overdispered. The convergence diagnostic statistic, known as the **estimated potential scale reduction (EPSR)**, is defined as $\sqrt{\hat{R}}$, where

$$\hat{R} = \frac{\hat{V}}{W} \quad (69)$$

$\hat{R}$ measures the degree to which the posterior variance would decrease if we were to continue sampling in the $S \to \infty$ limit. If $\hat{R} \approx 1$ for any given quantity such as y, then that estimate is reliable. Essentially it means the variance between the chains is similar to the variance within each chain.

Below is some simple code to compute $\hat{R}$.

```
function [Rhat, m, s] = EPSR(samples)
% EPSR "estimated potential scale reduction" statistic due to Gelman and Rubin.
%
% Inputs
% samples(i,j) for sample i, chain j
%
% Outputs
% Rhat = measure of scale reduction - value below 1.1 means converged:
% m = mean(samples)
% s = std(samples)

[n m] = size(samples);
meanPerChain = mean(samples,1); % each column of samples is a chain
meanOverall = mean(meanPerChain);
```

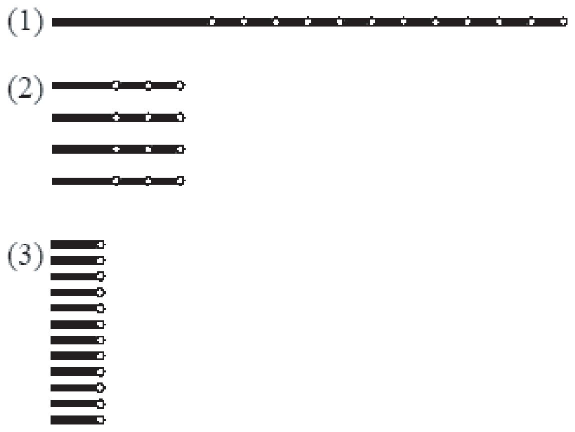


Figure 10: 3 possible sampling schemes for MCMC. Source: [Mac03].

```
% Rhat only works if more than one chain is specified.
if m > 1
  % between sequence variace
  B = (n/(m-1))*sum( (meanPerChain-meanOverall).^2);

  % within sequence variance
  varPerChain = var(samples);
  W = (1/m)*sum(varPerChain);

  vhat = ((n-1)/n)*W + (1/n)*B;
  Rhat = sqrt(vhat/(W+eps));
else
  Rhat = nan;
end
m = meanOverall;
s = std(samples(:));
```

$\hat{R}$ is widely used to assess convergence (e.g., in the BUGS software package). Since we are using means and variances, it is best to transform the scalar estimands to be approximately Normal (e.g., take logs of positive quantities and logits of quantities in 0..1).

Another practical question is how many chains to run. We could either run one long chain to ensure convergence, and then collect samples spaced far apart, or we could run many short chains, but that wastes the burnin time. In practice it is best to run a medium number (say 4) of medium length (say 10k-100k) chains and take samples from each. See Figure 10.

7 Advanced topics*

There are many topics we have not discussed. Some of the most important are

- Hybrid Monte Carlo (HMC), that exploits the gradient of $p(x)$ to move in the direction of high probability.
- Adapting the proposal distribution.
- Block updates, to update many variables at once.
- Rao-Blackwellisation, to integrate out many of the variables, thus reducing the size of the state space.
- Using data-driven (e.g., discriminative) proposal distributions [TZ02].

- Perfect sampling, in which samples are guaranteed to come from the target distribution, and there is no need to perform convergence diagnostics.
- Reversible jump MCMC, in which we can sample in spaces of different dimension.

For details, see e.g., [GRS96, RC04].

References

[AdFDJ03] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[Bis06] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006. Draft version 1.21.

[GRS96] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

[Mac03] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[RC04] C. Robert and G. Casella. *Monte Carlo Statisical Methods*. Springer, 2004. 2nd edition.

[TZ02] Z.W. Tu and S.C. Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.