Chapter 1

MATH5960/3871: Markov Chain Monte Carlo

1.1 Preliminaries

A *Markov chain* is a collection of random variables (possibly vectors in $\mathbb{R}^d$) $X_0, X_1, X_2, \ldots$ such that for any t, the distribution of $X_{t+1} \,|\, X_t$ is the same as the distribution of $X_{t+1} \,|\, (X_s,\ s \leq t)$. When the variables are discrete, we can write this as

MARKOV CHAIN

$$\mathbb{P}[X_t = x \,|\, X_0, \ldots, X_{t-1}] = \mathbb{P}[X_t = x \,|\, X_{t-1}] \,.$$

In other words, the distribution of the future X_{t+1} depends only on the present value X_t. Another way to state this is to say that, given the present value X_t, the future is independent of the past $X_0, X_1, \ldots, X_{t-1}$ — a property called the *Markov property*. We can thus talk about the *one-step transition density* $\kappa(x \,|\, X_t)$ of X_{t+1} written as $X_{t+1} \sim \kappa(x \,|\, X_t)$, so that

ONE-STEP TRANSITION DENSITY

$$\mathbb{P}[X_{t+1} \in A \,|\, X_t] = \int_A \kappa(y \,|\, X_t)\mathrm{d}y \,.$$

Note that here we assume that the chain is *time-homogeneous*, that is, the transition density is the same throughout all time/transitions.

If $X_0 \sim f_0(x)$ is an initial state from some *initial distribution*, then the joint pdf of $X_1, \ldots, X_t$ is

INITIAL DISTRIBUTION

$$\kappa(x_1 \,|\, x_0)\kappa(x_2 \,|\, x_1) \cdots \kappa(x_t \,|\, x_{t-1})$$

and the marginal pdf of X_t, given $X_0 = x_0$, is

$$\kappa_t(x \,|\, x_0) := \int \cdots \int \kappa(x_1 \,|\, x_0)\kappa(x_2 \,|\, x_1) \cdots \kappa(x_t \,|\, x_{t-1})\mathrm{d}x_1 \cdots \mathrm{d}x_{t-1} \,.$$

Note that we simply marginalized/integrated over x_1 to x_{t-1} above.

Suppose that π is some pdf (with corresponding distribution function Π) and the Markov chain satisfies the *global balance equations*

GLOBAL BALANCE EQUATIONS

$$\pi(x) = \int \kappa(x\,|\,y)\,\mathrm{d}\Pi(y) = \int \kappa(x\,|\,y)\,\pi(y)\mathrm{d}y\,.$$

Then, if the initial X_0 is drawn from π, every subsequent X_t for $t \geq 1$ will be distributed marginally according to π. To see this, note that

$$\begin{aligned}\mathbb{P}[X_1 \in A] &= \mathbb{E}[\mathbb{P}[X_1 \in A\,|\,X_0]] \\ &= \int \pi(x_0)\mathbb{P}[X_1 \in A\,|\,X_0 = x_0]\mathrm{d}x_0 \\ &= \int \pi(x_0) \int_A \kappa(x_1\,|\,x_0)\mathrm{d}x_1\mathrm{d}x_0 \\ &= \int_A \underbrace{\int \pi(x_0)\kappa(x_1\,|\,x_0)\mathrm{d}x_0}_{\pi(x_1)}\,\mathrm{d}x_1 = \int_A \pi(x_1)\mathrm{d}x_1,\end{aligned}$$

so that $X_1 \sim \pi$. Similarly, if $X_{t-1} \sim \pi$, then we have $X_t \sim \pi$, and therefore an induction argument shows that $X_t \sim \pi$ for all t. For this reason, we call the pdf π in the global balance equations the *stationary pdf*.

STATIONARY PDF

One of the main results for Markov chains is that under some technical assumptions to be explained shortly, X_t converges in distribution to $X \sim \pi$ as $t \uparrow \infty$. Such a chain is called *ergodic* with *limiting pdf* π.

LIMITING PDF

Instead of the global balance equations, one frequently imposes a stronger condition called the *detailed balance equations*

DETAILED BALANCE EQUATIONS

$$\pi(y)\kappa(x\,|\,y) = \pi(x)\kappa(y\,|\,x), \quad \text{for all valid } x, y\,.$$

These imply the global balance equations, but also impose a special kind of behavior on the Markov chain. Namely, if the detailed balance holds, then the behavior of the chain going forward in time $(X_t, X_{t+1}, \ldots)$ has the same statistical behavior as the chain in reverse time $(X_{t-1}, X_{t-2}, \ldots)$. Such a chain is called a *reversible chain*. If we take a video of a reversible chain and show the video to an independent observer, then the observer will not be able to tell if the video is played forwards or backwards.

REVERSIBLE CHAIN

A sufficient condition for a reversible chain to be ergodic is that it is also π-*irreducible*, that is, every valid state of the chain (those states x for which the stationary distribution $\pi(x) > 0$) can be reached from any other state in a finite number of steps. Mathematically, the chain is π-irreducible if

$$\pi(x)\pi(y) > 0 \Longrightarrow \kappa_t(x\,|\,y) > 0 \quad \text{for some finite } t\,.$$

If the chain $\{X_t, t \geq 0\}$ takes on discrete values (say $X \in \{1, 2, \ldots\}$), then we also need the additional assumption that the time to return to the same state is not a multiple of some integer ≥ 2. This last property is called *aperiodicity*. Mathematically, the chain is aperiodic if

$$\gcd\{n : \mathbb{P}[T_x = n\,|\,X_0 = x] = 1\} = 1$$

where $T_y = \inf_{n>0}\{n : X_n = y\}$ is the first time to visit state y after time zero. In other words, aperiodicity means that the greatest common divisor of the time (number of steps) to return to the same state of the chain is one.

Example 1.1 (Markov Chain Simulation) For time-homogeneous Markov chains with a discrete state space, we can visualize the one-step transitions by means of a *transition graph*, where arrows indicate possible transitions between states and the labels describe the corresponding probabilities. Figure 1.1 shows (on the left) the transition graph of the Markov chain $\{X_t, t = 0, 1, 2, \ldots\}$ with state space $\{1, 2, 3, 4\}$ and one-step transition matrix

TRANSITION GRAPH

$$\mathbf{P} = \begin{pmatrix} 0 & 0.2 & 0.5 & 0.3 \\ 0.5 & 0 & 0.5 & 0 \\ 0.3 & 0.7 & 0 & 0 \\ 0.1 & 0 & 0 & 0.9 \end{pmatrix}.$$

In the same figure (on the right) a typical outcome (path) of the Markov chain is shown. The path was simulated using the program below. In this implementation the Markov chain always starts at state 1.

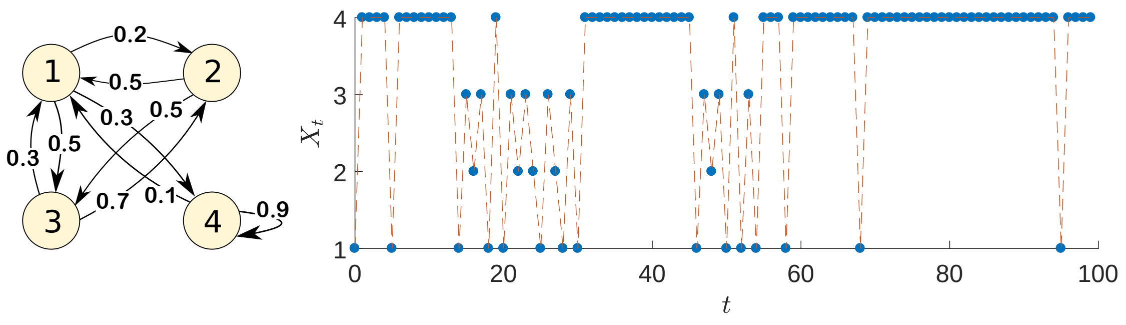

Figure 1.1: The transition graph (left) and a typical path (right) of the Markov chain.

```
% markovchain.m
n = 100;
P = [0,   0.2, 0.5, 0.3;
     0.5, 0,   0.5, 0;
     0.3, 0.7, 0,   0;
     0.1, 0,   0,   0.9];
x = zeros(1,n); x(1)= 1;
for t=1:n-1       % generate the Markov chain
    x(t+1) = min(find(cumsum(P(x(t),:))> rand));
end
hold on,  plot(0:n-1,x,'.','MarkerSize',20), plot(0:n-1,x)
hold off
```

1.2 Markov chain Monte Carlo

TARGET

Markov chain Monte Carlo (MCMC) is a generic method for *approximate* sampling from an arbitrary distribution π on $\mathscr{X} \subseteq \mathbb{R}^d$, usually referred to as the *target* distribution. The main idea is to evolve an ergodic Markov chain $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots$, where a state $\boldsymbol{X} \in \mathscr{X}$, whose limiting distribution is equal to the target distribution. Since the probability law of $\boldsymbol{X}_t$ converges to π, then $\boldsymbol{X}_t$ for large enough t may be used as an approximate draw from π. Conditioning on $\boldsymbol{X}_0 = \boldsymbol{x}_0$, this approximation can be stated in terms of the *total variation distance* converging to zero

TOTAL VARIATION DISTANCE

$$\sup_A |\kappa_t(A \,|\, \boldsymbol{x}_0) - \pi(A)| \to 0, \quad t \uparrow \infty \,.$$

Here $\pi(A)$ is shorthand notation for $\int_A \pi(\boldsymbol{x}) \mathrm{d}x$ and the $\sup_A$ means that we want the worst possible deviation of the approximation $\mathbb{P}[\boldsymbol{X}_t \in A]$ from the target $\pi[A]$.

As a consequence, to estimate an expectation $\mathbb{E}_\pi \phi(\boldsymbol{X})$, with $\boldsymbol{X} \sim \pi$, one can use the following ergodic estimator:

$$\frac{1}{t} \sum_{k=1}^{t} \phi(\boldsymbol{X}_k) \,. \tag{1.1}$$

Note that one can also use $\frac{1}{t+1} \sum_{k=0}^{t} \phi(\boldsymbol{X}_k)$. Since the $\boldsymbol{X}$'s are not exactly drawn from π, in general $\mathbb{E}\left[\frac{1}{t}\sum_{k=1}^{t} \phi(\boldsymbol{X}_k)\right] \neq \mathbb{E}_\pi \phi(\boldsymbol{X})$. Thus, we are interested in quantifying the magnitude of the bias

$$\left| \mathbb{E}\left[\frac{1}{t}\sum_{k=1}^{t} \phi(\boldsymbol{X}_k)\right] - \mathbb{E}_\pi \phi(\boldsymbol{X}) \right| \leq ?$$

It can be shown that for bounded functions ϕ (that is, $\|\phi\|_\infty \leq \infty$), the error is bounded by the total variation distance:

$$\left| \mathbb{E}\left[\frac{1}{t}\sum_{k=1}^{t} \phi(\boldsymbol{X}_k)\right] - \mathbb{E}_\pi \phi(\boldsymbol{X}) \right| \leq 2\|\phi\|_\infty \times \sup_A |\kappa_1(A \,|\, \boldsymbol{x}_0) - \pi(A)| \,.$$

Thus, if we have a chain $\{\boldsymbol{X}_t, t \geq 0\}$ and discard the first $b-1$ observations and compute the new estimator

$$\frac{1}{t-b+1} \sum_{k=b}^{t} \phi(\boldsymbol{X}_k)$$

we will have the bias:

$$\left| \mathbb{E}\left[\frac{1}{t-b+1}\sum_{k=b}^{t} \phi(\boldsymbol{X}_k)\right] - \mathbb{E}_\pi \phi(\boldsymbol{X}) \right| \leq 2\|\phi\|_\infty \times \sup_A |\kappa_b(A \,|\, \boldsymbol{x}_0) - \pi(A)|$$

BURN-IN

for any bounded function ϕ. We call these b wasted samples the *burn-in*. They are frequently discarded to ensure that the bias above is small.

It is clear that to make the bias small we want $\sup_A |\kappa_b(A \,|\, \boldsymbol{x}_0) - \pi(A)|$ to converge to zero very fast. In general, it is only possible to give a qualitative assessment of the convergence of the total variation distance. We say that the Markov chain is *geometrically ergodic* (converges exponentially fast) if there

GEOMETRICALLY ERGODIC

exist constants $c_1(\boldsymbol{x}_0) > 0$ (possibly dependent on the initial $\boldsymbol{x}_0$) and $r \in (0,1)$ such that for all t we have

$$\sup_A |\kappa_t(A \,|\, \boldsymbol{x}_0) - \pi(A)| \leq c_1(\boldsymbol{x}_0) r^t \,.$$

In addition, if the above constant $c_1(\boldsymbol{x}_0)$ is bounded for all $\boldsymbol{x}_0$, that is, $\sup_{\boldsymbol{x}_0} c_1(\boldsymbol{x}_0) < \infty$, then we say that the Markov chain is *uniformly geometrically ergodic*.

In addition to the bias, we also want (1.1) to have small variance. Since the $\boldsymbol{X}$'s in the Markov chain are dependent, we cannot easily estimate the variance of (1.1) (even after discarding some sampels as burn-in). For example, the formula for the variance of a sum of independent random variables does not apply:

$$\mathbb{V}\left(\frac{1}{t}\sum_{k=1}^{t} \phi(\boldsymbol{X}_k)\right) \neq \frac{\mathbb{V}(\phi(\boldsymbol{X}_1))}{t} \,.$$

One way around this obstacle is to apply *thinning* to the chain. This means that we select an integer b, preferably a divisor of t, so that $t = m \times b$ for some integer m. Then, pick every b-th sample in the sequence: THINNING

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{b-1}, \underbrace{\boldsymbol{X}_b}_{\boldsymbol{X}'_1}, \boldsymbol{X}_{b+1}, \ldots, \boldsymbol{X}_{2b-1}, \underbrace{\boldsymbol{X}_{2b}}_{\boldsymbol{X}'_2}, \boldsymbol{X}_{2b+1}, \ldots \boldsymbol{X}_{mb-1}, \underbrace{\boldsymbol{X}_{mb}}_{\boldsymbol{X}'_m} \,.$$

The resulting sequence $\boldsymbol{X}'_1, \ldots, \boldsymbol{X}'_m$ is then approximately independent and we can approximate the variance of (1.1) via

$$\frac{b \times \frac{1}{m}\sum_{k=1}^{m}(Z_k - \bar{Z}_m)^2}{t} \,,$$

where $Z_k := \phi(\boldsymbol{X}'_k)$. Sometimes a more accurate estimate can be obtained by instead using

$$Z_k := \frac{1}{b} \sum_{j=(k-1)b+1}^{kb} \phi(\boldsymbol{X}_j), \quad k = 1, \ldots, m \,. \tag{1.2}$$

The variance estimator using this last definition of Z_k is called the *batch-means* variance estimator. We will revisit this issue later on. BATCH-MEANS

1.3 Metropolis–Hastings Algorithm

Suppose that we wish to generate samples from an arbitrary multidimensional pdf

$$\pi(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{\mathcal{Z}}, \quad \boldsymbol{x} \in \mathscr{X} \,,$$

where $f(\boldsymbol{x})$ is a known positive function and $\mathcal{Z}$ is a known or unknown normalizing constant. Let $g(\boldsymbol{y} \,|\, \boldsymbol{x})$ be a *proposal* or *instrumental* density: a density describing how to go from state $\boldsymbol{x}$ to $\boldsymbol{y}$. Similar to the acceptance–rejection method for simulating random variables, the Metropolis–Hastings algorithm [4] is based on the following "trial-and-error" strategy. PROPOSAL

Algorithm 1.1 (Metropolis–Hastings Algorithm)

```
Require: Some $\boldsymbol{X}_0$ for which $f(\boldsymbol{X}_0) > 0$.
  for $k = 0,\ldots,t-1$ do
      Given the current state $\boldsymbol{X}_k$, generate $\boldsymbol{Y} \sim g(\boldsymbol{y}\,|\,\boldsymbol{X}_k)$.
      Simulate $U \sim \mathsf{U}(0,1)$ and set
          $$\boldsymbol{X}_{k+1} = \begin{cases} \boldsymbol{Y} & \text{if } U \leq \alpha(\boldsymbol{X}_k,\boldsymbol{Y}) \\ \boldsymbol{X}_k & \text{otherwise}, \end{cases} \qquad (1.3)$$
      where
          $$\alpha(\boldsymbol{x},\boldsymbol{y}) := \min\left\{\frac{f(\boldsymbol{y})\,g(\boldsymbol{x}\,|\,\boldsymbol{y})}{f(\boldsymbol{x})\,g(\boldsymbol{y}\,|\,\boldsymbol{x})}, 1\right\} \qquad (1.4)$$
      is the acceptance probability.
  return $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$
```

ACCEPTANCE PROBABILITY

METROPOLIS–HASTINGS CHAIN

We thus obtain the so-called *Metropolis–Hastings chain*, $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$, with $\boldsymbol{X}_t$ approximately distributed according to π for large t. The original Metropolis algorithm [4] is suggested for symmetric proposal functions; that is, for $g(\boldsymbol{y}\,|\,\boldsymbol{x}) = g(\boldsymbol{x}\,|\,\boldsymbol{y})$. Hastings [3] modified the original MCMC algorithm to allow nonsymmetric proposal functions, hence the name Metropolis–Hastings algorithm.

In the discrete case, a single step of the Metropolis-Hastings sampler follows the Markov transition density:

$$\kappa(\boldsymbol{y}\,|\,\boldsymbol{x}) = \begin{cases} g(\boldsymbol{y}\,|\,\boldsymbol{x})\,\alpha(\boldsymbol{x},\boldsymbol{y}), & \text{if } \boldsymbol{y} \neq \boldsymbol{x} \\ 1 - \sum_{\boldsymbol{z}\neq\boldsymbol{x}} g(\boldsymbol{z}\,|\,\boldsymbol{x})\,\alpha(\boldsymbol{x},\boldsymbol{z}), & \text{if } \boldsymbol{y} = \boldsymbol{x}\,. \end{cases} \qquad (1.5)$$

It is easy to show that this transition density satisfies the detailed balance equation. We need to show that

$$f(\boldsymbol{x})\,\kappa(\boldsymbol{y}\,|\,\boldsymbol{x}) = f(\boldsymbol{y})\,\kappa(\boldsymbol{x}\,|\,\boldsymbol{y}) \quad \text{for all } \boldsymbol{x},\boldsymbol{y}\,. \qquad (1.6)$$

We thus need to check the detailed balance for three cases: (1) $\boldsymbol{x} = \boldsymbol{y}$, (2) $\boldsymbol{x} \neq \boldsymbol{y}$ and $f(\boldsymbol{y})g(\boldsymbol{x}\,|\,\boldsymbol{y}) \leq f(\boldsymbol{x})g(\boldsymbol{y}\,|\,\boldsymbol{x})$, and (3) $\boldsymbol{x} \neq \boldsymbol{y}$ and $f(\boldsymbol{y})g(\boldsymbol{x}\,|\,\boldsymbol{y}) > f(\boldsymbol{x})g(\boldsymbol{y}\,|\,\boldsymbol{x})$.

Case (1) holds trivially. For case (2), $\alpha(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{y})g(\boldsymbol{x}\,|\,\boldsymbol{y})/(f(\boldsymbol{x})g(\boldsymbol{y}\,|\,\boldsymbol{x}))$ and $\alpha(\boldsymbol{y},\boldsymbol{x}) = 1$. Consequently, $\kappa(\boldsymbol{y}\,|\,\boldsymbol{x}) = f(\boldsymbol{y})g(\boldsymbol{x}\,|\,\boldsymbol{y})/f(\boldsymbol{x})$ and $\kappa(\boldsymbol{x}\,|\,\boldsymbol{y}) = g(\boldsymbol{x}\,|\,\boldsymbol{y})$, so that (1.6) holds. Similarly, for case (3), $\alpha(\boldsymbol{x},\boldsymbol{y}) = 1$ and $\alpha(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x})g(\boldsymbol{y}\,|\,\boldsymbol{x})/(f(\boldsymbol{y})g(\boldsymbol{x}\,|\,\boldsymbol{y}))$. Consequently, $\kappa(\boldsymbol{y}\,|\,\boldsymbol{x}) = g(\boldsymbol{y}\,|\,\boldsymbol{x})$ and $\kappa(\boldsymbol{x}\,|\,\boldsymbol{y}) = f(\boldsymbol{x})g(\boldsymbol{y}\,|\,\boldsymbol{x})/f(\boldsymbol{y})$, so that (1.6) holds again.

In general, we can write the transition density as a mixture density of the form:

$$\kappa(\boldsymbol{y}\,|\,\boldsymbol{x}) = \frac{g(\boldsymbol{y}\,|\,\boldsymbol{x})\alpha(\boldsymbol{y},\boldsymbol{x})}{\alpha^*(\boldsymbol{x})}\alpha^*(\boldsymbol{x}) + (1-\alpha^*(\boldsymbol{x}))\delta(\boldsymbol{y}-\boldsymbol{x}_k),$$

where

$$\alpha^*(\boldsymbol{x}) := \int g(\boldsymbol{y}\,|\,\boldsymbol{x})\alpha(\boldsymbol{y},\boldsymbol{x})\mathrm{d}\boldsymbol{y}$$

is the mixture weight indicating the probability that we draw from the density of the first component: $\frac{g(\boldsymbol{y}\,|\,\boldsymbol{x})\alpha(\boldsymbol{y},\boldsymbol{x})}{\alpha^*(\boldsymbol{x})}$.

1.3.1 Random-Walk Sampler

A special case of the Metropolis-Hastings sampler is when the proposal density is symmetric, that is, $g(\boldsymbol{x}\,|\,\boldsymbol{y}) = g(\boldsymbol{y}\,|\,\boldsymbol{x})$. Then, the acceptance rate (1.4) simplifies to

$$\alpha(\boldsymbol{x},\boldsymbol{y}) = \min\left\{\frac{f(\boldsymbol{y})}{f(\boldsymbol{x})}, 1\right\} .$$

RANDOM-WALK SAMPLER

This version of the Metropolis-Hastings sampler is called the *random-walk sampler*. An example of a random-walk sampler is when the proposal $g(\boldsymbol{x}\,|\,\boldsymbol{y})$ corresponds to the $\mathsf{N}(\boldsymbol{y}, \varsigma^2\mathbf{I})$ for some scale parameter $\varsigma > 0$. When we wish to have tails heavier than the Gaussian ones, we can instead use a proposal pdf $g(\boldsymbol{x}\,|\,\boldsymbol{y})$ corresponding to the multivariate student-t distribution $\mathsf{t}_\nu(\boldsymbol{y}, \varsigma^2\mathbf{I})$ with $\nu > 0$ degrees of freedom.

■ **Example 1.2 (Random Walk Sampler)** Consider the two-dimensional pdf

$$\pi(x_1,x_2) = \frac{\mathrm{e}^{-\frac{1}{4}\sqrt{x_1^2+x_2^2}}\left(\sin\left(2\sqrt{x_1^2+x_2^2}\right)+1\right)}{\mathcal{Z}}, \quad -2\pi < x_1 < 2\pi,\ -2\pi < x_2 < 2\pi, \tag{1.7}$$

where $\mathcal{Z}$ is an unknown normalization constant. The graph of this pdf (unnormalized) is depicted in the left panel of Figure 1.2.

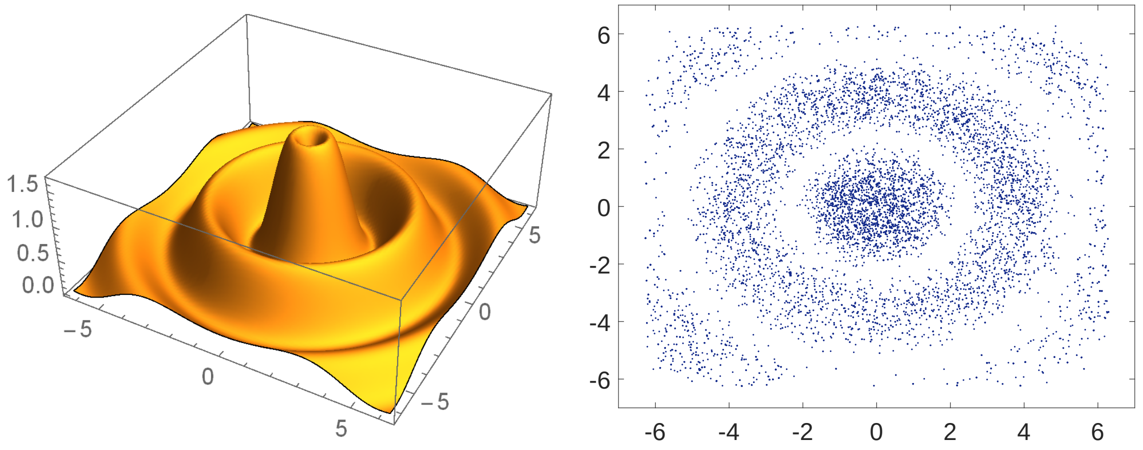

Figure 1.2: Points from the random walk sampler (right panel) are approximately distributed according to the two-dimensional target pdf in the left panel.

The following MATLAB program implements a random walk sampler to (approximately) draw $N = 10^4$ dependent samples from the pdf f. At each step, given a current state $\boldsymbol{x}$, a proposal $\boldsymbol{Y}$ is drawn from the $\mathsf{N}(\boldsymbol{x}, \mathbf{I})$ distribution. That is, $\boldsymbol{Y} = \boldsymbol{x} + \boldsymbol{Z}$, with $\boldsymbol{Z}$ bivariate standard normal. We see in the right panel of Figure 1.2 that the sampler works correctly. The starting point for the Markov chain is chosen as $(0,0)$. Note that the normalization constant c is not used in the program.

```
% rwsamp1.m
f=@(X1,X2) exp(-sqrt(X1.^2 + X2.^2)/4).*...
(sin(2*sqrt(X1.^2 + X2.^2)) + 1).*(-2*pi<X1).*(X1<2*pi).*(-2*pi<X2).*(X2<2*pi);
N = 100000; %sample size
xx = zeros(N,2); x = [0,0]; xx(1,:) = x;
for i=2:N
    y = x + randn(1,2); %proposal
    alpha = min(f(y(1),y(2))/f(x(1),x(2)),1); %acceptance probab.
    r = (rand < alpha);
    x = r*y + (1-r)*x; %next value of the Markov chain
    xx(i,:) = x;
end
plot(xx(:,1),xx(:,2),'.')
```

1.3.2 Independence Sampler

Another special case of the Metropolis-Hastings sampler is when the proposal density is independent of the current state of the chain, that is, $g(\boldsymbol{x}\,|\,\boldsymbol{y}) = g(\boldsymbol{x})$. Then, the acceptance rate (1.4) becomes

$$\alpha(\boldsymbol{x},\boldsymbol{y}) = \min\left\{\frac{f(\boldsymbol{y})g(\boldsymbol{x})}{f(\boldsymbol{x})g(\boldsymbol{y})},1\right\} .$$

INDEPEN-DENCE SAMPLER

Such a sampler is called an *independence sampler*.

■ **Example 1.3 (Independence Sampler)** Consider again the target (1.7) and suppose we use the proposal

$$g(\boldsymbol{x}) \propto \exp(-|x_1|/4 - |x_2|/4)$$

Simulating from g is equivalent to sampling independently from $\mathsf{Laplace}(0,4)$ distributions. The $\mathsf{Laplace}(\mu,\sigma)$ density is of the form:

$$\exp\left(-\frac{|x-\mu|}{\sigma}\right)/(2\sigma) .$$

Note that (by the inverse transform method) if $X = \sigma \mathrm{sign}(U)\ln(1-2|U|)$, where $U \sim \mathsf{U}(-1/2,1/2)$, then $X \sim \mathsf{Laplace}(\mu,\sigma)$. Running the following MATLAB code will give us the same as the right panel of Figure 1.2.

```
f=@(X1,X2) exp(-sqrt(X1.^2 + X2.^2)/4).*...
(sin(2*sqrt(X1.^2 + X2.^2)) + 1).*(-2*pi<X1).*(X1<2*pi).*(-2*pi<X2).*(X2<2*pi);
g=@(x,s)(exp(-abs(x)/s)/(2*s));
N = 10^5; %sample size
xx = zeros(N,2); x = [0,0]; xx(1,:) = x;
```

```
s=4;
for i=2:N
    U=rand(1,2)-1/2;
    y = s*sign(U).*log(1-2*abs(U)); %proposal
    alpha = min(f(y(1),y(2))/f(x(1),x(2))*...
        g(x(1),s)*g(x(2),s)/(g(y(1),s)*g(y(2),s)),1); %acceptance probab.
    r = (rand < alpha);
    x = r*y + (1-r)*x; %next value of the Markov chain
    xx(i,:) = x;
end
plot(xx(:,1),xx(:,2),'.')
```

In general, it is difficult to say which sampler is better. One should try to validate the output of one sampler by running other viable samplers and if the results agree then the confidence in the approximation increases.

Nevertheless, it is common to assess the quality of the sampler by looking at an autocorrelation plot of some function of the output. For example, we can choose to plot the (estimated) autocorrelation function of $\{X_{k,1}, k = 1, 2, \ldots\}$. The estimate of the autocorrelation of $\{Z_k, k = 1, \ldots, t\}$ is

$$\hat{r}(j) := \frac{1}{t-j-1}\sum_{k=1}^{t-j}(X_k - \bar{X})(X_{k+j} - \bar{X}), \quad \bar{X} := \frac{1}{t}\sum_{k=1}^{t} X_k$$

for $j < t$, where j is called the *autocorrelation lag*. The autocorrelation heuristic (rule of thumb) says is that the faster the autocorrelation function decays to zero, the better the MCMC algorithm.

AUTOCOR-RELATION LAG

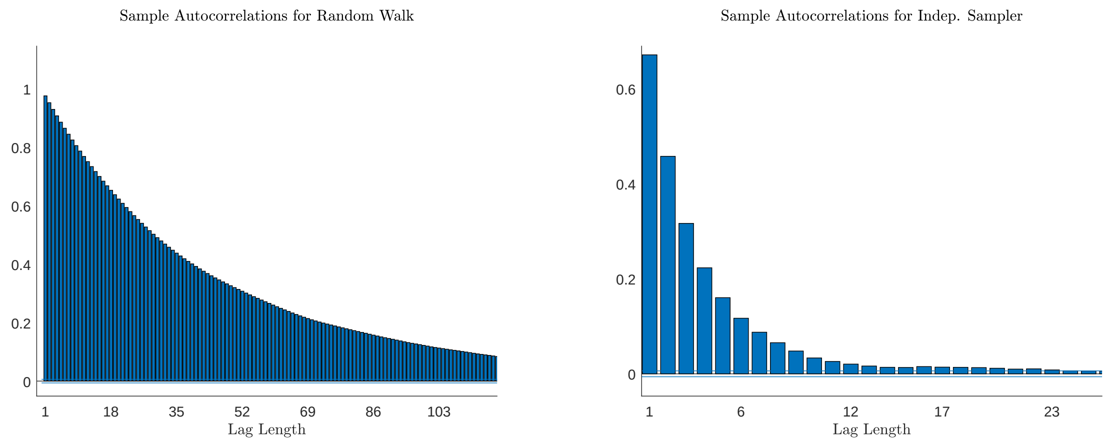

Figure 1.3: A comparison between the autocorrelation plots of the two samplers.

For the independence sampler it is possible to give a theoretical estimate of the convergence to the target pdf π. This is given in the following.

Useful Result 1.1: Convergence of the Independence Sampler

Suppose that there is a constant c such that for all $\boldsymbol{x}$ the likelihood ratio is bounded:

$$W(\boldsymbol{x}) := f(\boldsymbol{x})/g(\boldsymbol{x}) \leq c\,.$$

Then, the sampler is geometrically ergodic with total variation distance:

$$\sup_A |\kappa_t(A \,|\, \boldsymbol{x}_0) - \pi(A)| \leq 2\,(1 - \mathbb{E}_g W(\boldsymbol{X})/c)^t\ ,$$

where, of course, $\mathbb{E}_g W(\boldsymbol{X}) = \int f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. On the other hand, if such a constant does not exist, then the sampler does is not geometrically ergodic (and may even fail to converge).

■ **Example 1.4 (Independence Sampler Continued)** *Consider again the proposal* $g(\boldsymbol{x}) \propto \exp(-|x_1|/4 - |x_2|/4)$ *for the target* (1.7) *and note that*

$$\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \propto \exp\left(-\frac{\sqrt{x_1^2+x_2^2} - |x_1| - |x_2|}{4}\right) \mathbb{I}\{\boldsymbol{x} \in [-2\pi, 2\pi]^2\},$$

which is bounded due to the indicator function. If the target was extended to an unbounded domain ($\boldsymbol{x} \in \mathbb{R}^2$), then this independence sampler would not be converging, because $x_1^2 + x_2^2 \leq (|x_1| + |x_2|)^2$ and hence the ratio f/g would be unbounded.

1.4 Gibbs Sampler

The *Gibbs sampler* can be viewed as a particular instance of the Metropolis–Hastings algorithm for generating d-dimensional random vectors [2]. Due to its importance it is presented separately. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed from a sequence of conditional distributions, in either a deterministic or random fashion.

Suppose that we wish to sample a random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ according to a target pdf $\pi(\boldsymbol{x})$. Let $\pi(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d$.

Algorithm 1.2 (Gibbs Sampler)

```
Require: an initial state $\boldsymbol{X}_0$
  for $k = 0, 1, \ldots, t-1$ do
      Set $\boldsymbol{x} \leftarrow \boldsymbol{X}_k$.
      Draw $Y_1$ from the conditional pdf $\pi(x_1 \,|\, x_2, \ldots, x_d)$.
      for $i = 2, \ldots, d-1$ do
          Draw $Y_i$ from $\pi(x_i \,|\, Y_1, \ldots, Y_{i-1}, x_{i+1}, \ldots, x_d)$.
      Draw $Y_d$ from $\pi(x_d \,|\, Y_1, \ldots, Y_{d-1})$.
      Set $\boldsymbol{X}_{k+1} \leftarrow (Y_1, \ldots, Y_d)^\top$
  return $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$
```

The transition pdf is given by

$$\kappa_{1\to d}(\boldsymbol{y}\,|\,\boldsymbol{x}) = \prod_{i=1}^{d} \pi(y_i\,|\,y_1,\ldots,y_{i-1},x_{i+1},\ldots,x_d)\ , \tag{1.8}$$

where the subscript $1 \to d$ indicates that the components of vector $\boldsymbol{X}$ are updated in the order $1 \to 2 \to 3 \to \cdots \to d$. Note that in the Gibbs sampler *every* "proposal" $\boldsymbol{y}$, is accepted. The transition density of the reverse move $\boldsymbol{y} \to \boldsymbol{X}$, in which the vector $\boldsymbol{y}$ is updated in the order $d \to d-1 \to d-2 \to \cdots \to 1$ is

$$\kappa_{d\to 1}(\boldsymbol{x}\,|\,\boldsymbol{y}) = \prod_{i=1}^{d} \pi(x_i\,|\,y_1,\ldots,y_{i-1},x_{i+1},\ldots,x_d)\ .$$

The reason the sampler works is that the transition density satisfies the global balance equations as shown by the following result.

Useful Result 1.2: Hammersley-Clifford

Let $\pi(x_i)$ be the i-th marginal density of $\pi(\boldsymbol{x})$ and suppose that π satisfies the *positivity condition*, that is, $\prod_i \pi(x_i) > 0 \Longrightarrow \pi(\boldsymbol{x}) > 0$ for all $\boldsymbol{x}$. Then,

POSITIVITY CONDITION

$$\pi(\boldsymbol{y})\kappa_{d\to 1}(\boldsymbol{x}\,|\,\boldsymbol{y}) = \pi(\boldsymbol{x})\kappa_{1\to d}(\boldsymbol{y}\,|\,\boldsymbol{x}),$$

the transition density $\kappa_{d\to 1}$ satisfies the global balance equation: $\int \pi(\boldsymbol{x})\kappa_{1\to d}(\boldsymbol{y}\,|\,\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \pi(\boldsymbol{y})$, and the Gibbs sampler Markov chain is irreducible with limiting pdf π.

■ **Example 1.5 (Failure of Positivity)** *Consider simulating from the bivariate density:*

$$\pi(x,y) = 1/2, \quad (x,y) \in [0,1]^2 \cup [2,3]^2\ .$$

Then, $X\,|\,Y = y \sim \mathsf{U}(0,1)$ if $y \in [0,1]$ and $X\,|\,Y = y \sim \mathsf{U}(2,3)$ if $y \in [2,3]$. Similarly for $Y\,|\,X = x$, because we have a symmetry between x and y. Note that once we start with $Y = y \in [0,1]$ we never exit the set $[0,1]^2$. Similarly, if we start with $Y = y \in [2,3]$ we never exit the set $[2,3]^2$. Thus, the chain is not irreducible. No matter how much we run the Markov chain, the initial state is not forgotten! Formally we can check that the positivity condition is not satisfied. Marginally, $X,Y \sim \mathsf{U}([0,1]\cup[2,3])$, so that we can find an $x' \in [0,1]$ and $y' \in [2,3]$ such that $\pi(x')\pi(y') > 0$, but $\pi(x',y') = 0$.

Algorithm 1.2 presents a *systematic* (coordinatewise) Gibbs sampler. That is, the components of vector $\boldsymbol{X}$ are updated in the coordinatewise order $1 \to 2 \to \cdots \to d$. The completion of all the conditional sampling steps in the specified order is called a *cycle*. Alternative updating of the components of vector $\boldsymbol{X}$ are possible. In the *reversible Gibbs sampler* a single cycle consists of the coordinatewise updating

RE-VERSIBLE GIBBS SAMPLER

$$1 \to 2 \to \cdots \to d-1 \to d \to d-1 \to \cdots \to 2 \to 1\ .$$

In the *random scan Gibbs sampler* a single cycle can either consist of one or several coordinates selected uniformly from the integers $1,\ldots,d$, or a random permutation $p_1 \to p_2 \to \cdots \to p_d$ of all coordinates.

RANDOM SCAN GIBBS SAMPLER

In all cases, except for the systematic Gibbs sampler, the resulting Markov chain $\{\boldsymbol{X}_t, t = 1, 2, \ldots\}$ is *reversible*.

■ **Example 1.6 (Gibbs sampler for the Bayesian normal model)** Gibbs samplers are often applied in Bayesian statistics, to sample from the posterior pdf. Consider for instance the Bayesian normal model

$$\pi(\mu, \sigma^2) = 1/\sigma^2$$
$$(\boldsymbol{X} \mid \mu, \sigma^2) \sim \mathsf{N}(\mu \mathbf{1}, \sigma^2 \mathbf{I}) .$$

We know that the posterior pdf is:

$$\pi(\mu, \sigma^2 \mid \tau) \propto \left(\sigma^2\right)^{-n/2-1} \exp\left\{ -\frac{1}{2} \frac{\sum_i (x_i - \mu)^2}{\sigma^2} \right\} . \tag{1.9}$$

To simulate samples μ and σ^2 from from (1.9) using the Gibbs sampler, we need the distributions of both $(\mu \mid \sigma^2, \tau)$ and $(\sigma^2 \mid \mu, \tau)$. To find $\pi(\mu \mid \sigma^2, \tau)$, view the right-hand side of (1.9) as a function of μ only, regarding σ^2 as a constant. This gives

$$\pi(\mu \mid \sigma^2, \tau) \propto \exp\left\{ -\frac{n\mu^2 - 2\mu \sum_i x_i}{2\sigma^2} \right\} = \exp\left\{ -\frac{\mu^2 - 2\mu\bar{x}}{2(\sigma^2/n)} \right\}$$
$$\propto \exp\left\{ -\frac{1}{2} \frac{(\mu - \bar{x}_n)^2}{\sigma^2/n} \right\} . \tag{1.10}$$

This shows that $(\mu \mid \sigma^2, \tau)$ has a normal distribution with mean $\bar{x}_n$ and variance σ^2/n.

Similarly, to find $\pi(\sigma^2 \mid \mu, \tau)$, view (1.9) as a function of σ^2, regarding μ as a constant. This gives

$$\pi(\sigma^2 \mid \mu, \tau) \propto (\sigma^2)^{-n/2-1} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2 / \sigma^2 \right\} , \tag{1.11}$$

showing that $(\sigma^2 \mid \mu, \tau)$ has an inverse-gamma distribution with parameters $n/2$ and $\Sigma_{i=1}^n (x_i - \mu)^2/2$. The Gibbs sampler thus involves the repeated simulation of

$$(\mu \mid \sigma^2, \tau) \sim \mathsf{N}\left(\bar{x}_n, \, \sigma^2/n\right) \quad \text{and} \quad (\sigma^2 \mid \mu, \tau) \sim \mathsf{Inv-Gamma}\left(n/2, \, \textstyle\sum_{i=1}^n (x_i - \mu)^2/2\right).$$

Simulation $X \sim \mathsf{Inv-Gamma}(\alpha, \lambda)$ is achieved by first generating $Z \sim \mathsf{Gamma}(\alpha, \lambda)$ and then returning $X = 1/Z$.

The following MATLAB script defines a small data set of size $n = 10$ (which was randomly simulated from a standard normal distribution), and implements the systematic Gibbs sampler to simulate from the posterior distribution, using $N = 10^5$ samples.

```
% gibbs_norm_bayes.m
x = [-0.9472, 0.5401, -0.2166, 1.1890,1.3170,...
     -0.4056, -0.4449, 1.3284  0.8338, 0.6044];
n = size(x,2);
```

```
sample_mean = mean(x)
sample_var = var(x)
sig2 = var(x); mu = sample_mean; %initial state
N = 10^5; %sample size for Gibbs sampler
gibbs_sample = zeros(N,2);
for k=1:N
    mu = sample_mean + sqrt(sig2/n)*randn;  %draw mu
    V = sum((x -mu).^2)/2;
    sig2 = 1/gamrnd(n/2,1/V); %draw sigma^2
    gibbs_sample(k,:) = [mu,sig2];
end
hold on
plot(gibbs_sample(:,1),gibbs_sample(:,2),'.')
plot(mean(x),var(x),'wo')
hold off
```

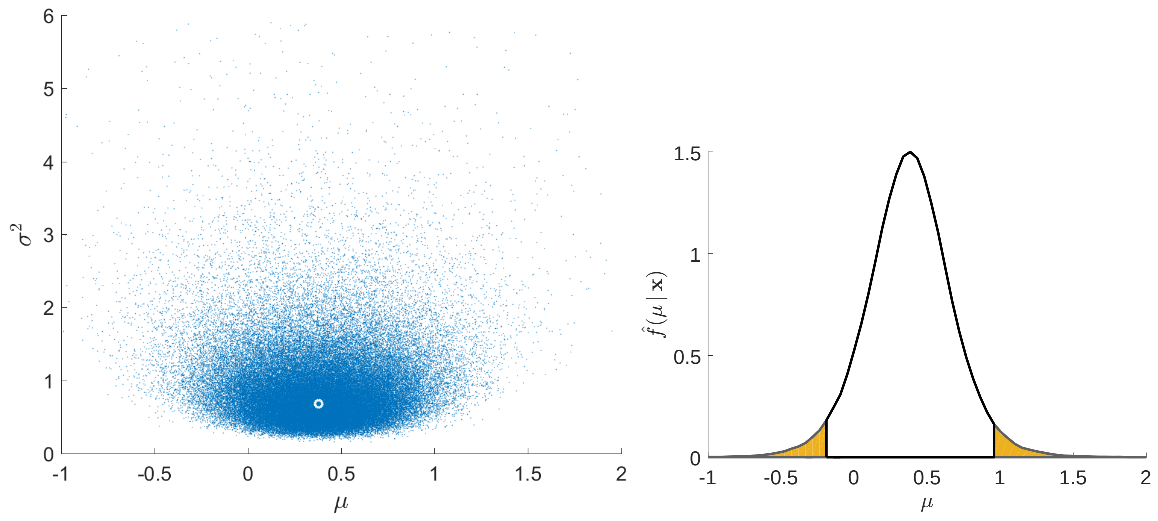

Figure 1.4: Left: approximate draws from the posterior pdf $\pi(\mu,\sigma^2 \mid \tau)$ obtained via the Gibbs sampler. Right: estimate of the posterior pdf $\pi(\mu \mid \tau)$.

The left panel of Figure 1.4 shows the (μ,σ^2) points generated by the Gibbs sampler. Also shown, via the white circle, is the point $(\bar{x},s^2)$, where $\bar{x} = 0.3798$ is the sample mean and $s^2 = 0.6810$ the sample variance. This posterior point cloud visualizes the considerable uncertainty in the estimates. By projecting the (μ,σ^2) points onto the μ-axes — that is, by ignoring the σ^2 values — one obtains (approximate) samples from the posterior pdf of μ; that is, $\pi(\mu \mid \tau)$. The right panel of Figure 1.4 shows a kernel density estimate of this pdf. The corresponding 0.05 and 0.95 sample quantiles were found to be -0.2054 and 0.9662, respectively. The 95% credible interval for μ is thus $(-0.2054, 0.9662)$, which contains the true expectation 0. Similarly, an estimated 95% credible interval for σ^2 is $(0.3218, 2.2485)$, which contains the true variance 1.

■ **Example 1.7 (Sampling from Truncated Multivariate Normal)** *Suppose we wish to simulate from the bivariate normal density $\boldsymbol{X} \sim \mathsf{N}(\mathbf{o}, \Sigma)$ conditional on $\boldsymbol{X} \geq \mathbf{o}$. In other words, the target is*

$$\pi(\boldsymbol{x}) \propto \exp(-\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}/2)\mathbb{I}\{\boldsymbol{x} \geq \mathbf{o}\} \,,$$

where we may assume (without loss of generality) that Σ is a correlation matrix:

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} .$$

Let $\mathsf{TN}_{\mathcal{I}}(\mu, \sigma^2)$ be notation for the distribution of $X \sim \mathsf{N}(\mu, \sigma^2)$, conditional on $X \in \mathcal{I}$. The conditional densities are $X \,|\, Y = y \sim \mathsf{TN}_{[0,\infty)}(\rho y, 1-\rho^2)$ and $Y \,|\, X = x \sim \mathsf{TN}_{[0,\infty)}(\rho x, 1-\rho^2)$, which are simple to simulate from. Figure 1.5 shows the output of this sampler using 10^4 iterations, starting with $X_1 = 0$.

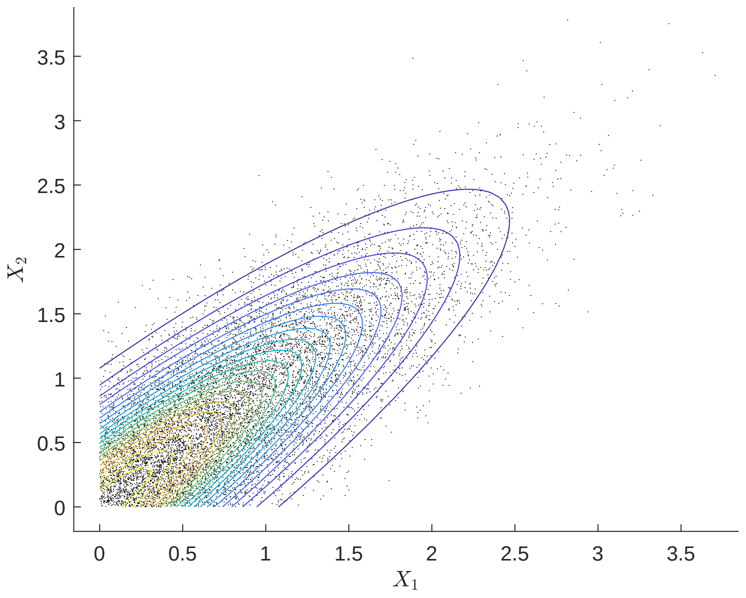

Figure 1.5: Contours of the target π and (approximate) random realizations from π using Gibbs sampling.

We can make a transformation to possibly speed up the convergence of the sampler. Let $(x,y)^\top =$

$\mathbf{L}\boldsymbol{z}$, *where*

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{bmatrix} .$$

Then,

$$\pi(\boldsymbol{z}) \propto \exp(-\|\boldsymbol{z}\|^2/2)\mathbb{I}\{\mathbf{L}\boldsymbol{z} \geq \mathbf{o}\} .$$

The first conditional for Gibbs sampling becomes $Z_1 \,|\, Z_2 = z_2 \sim \mathsf{TN}_{[\alpha,\infty)}(0,1)$, *where*

$$\alpha = \max\left\{0, -\sqrt{1-\rho^2}z_2/\rho\right\},$$

assuming $\rho > 0$. *The second conditional becomes* $Z_2 \,|\, Z_1 = z_1 \sim \mathsf{TN}_{[\alpha,\infty)}(0,1)$ *with* $\alpha = -z_1\rho/\sqrt{1-\rho^2}$. *Once we have simulated* $\boldsymbol{Z} = (Z_1, Z_2)^\top$ *with (approximate) pdf* $\pi(\boldsymbol{z})$, *we can obtain an* $\boldsymbol{X}$ *from* $\pi(\boldsymbol{x})$ *via the back-transformation* $\boldsymbol{X} = \mathbf{L}\boldsymbol{Z}$. *Figure 1.6 suggests that this transformation is effective in accelerating the convergence of the Gibbs sampler. Note that the figures show an error bars (horizonal lines) such that anything below the error bars may be considered to be zero.*

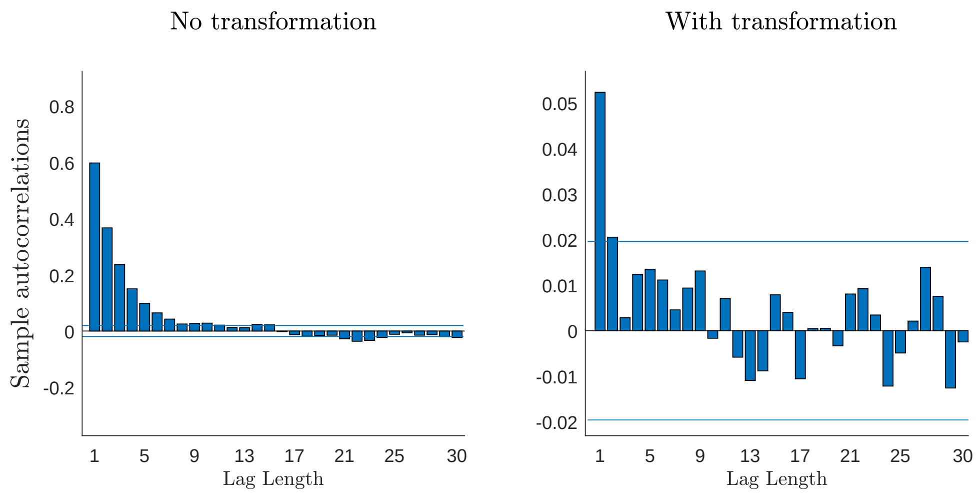

Figure 1.6: Left panel: autocorrelations of X_1 for Gibbs sampling without the transformation. Right panel: autocorrelations with the transformation.

1.4.1 Grouping and Blocking

Gibbs sampling is advantageous whenever it is easy to sample from the conditional distributions of the joint density. Note that it is not necessary to update each component of the random vector $\boldsymbol{X}$ individually. Instead, blocks or groups of variables can be updated simultaneously. For example, to sample from the joint pdf $\pi(x_1, x_2, x_3)$ we can consider the following version of the Gibbs sampler.

Algorithm 1.3 (Grouped Gibbs Sampling from $\pi(x_1,x_2,x_3)$)

```
Require: an initial state $\boldsymbol{X}_0$
  for $k = 0,1,\ldots,t-1$ do
      Set $(x_1,x_2,x_3) \leftarrow \boldsymbol{X}_k$.
      Draw $(Y_1,Y_2)$ from the conditional pdf $\pi(y_1,y_2 \,|\, x_3)$.
      Draw $Y_3$ from the conditional pdf $\pi(y_3 \,|\, Y_1,Y_2)$.
      Set $\boldsymbol{X}_{k+1} \leftarrow (Y_1,Y_2,Y_3)$
  return $\boldsymbol{X}_0,\boldsymbol{X}_1,\ldots,\boldsymbol{X}_t$
```

The grouped variables in Algorithm 1.3 are x_1 and x_2. Significant speed-up of the convergence of the chain can be achieved when highly correlated variables are grouped together [5]. An example of grouping is given in the next section.

1.4.2 Auxiliary Variables

Auxiliary variable tricks exploit the fact that every density $\pi(\boldsymbol{x})$ can be viewed as a marginal density:

$$\pi(\boldsymbol{x}) = \int \pi(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}\,,$$

AUXILIARY/LATENT VARIABLE

where $\pi(\boldsymbol{x},\boldsymbol{y})$ is the joint density of random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Here, $\boldsymbol{Y}$ is called an *auxiliary/latent variable*. The vector $(\boldsymbol{X},\boldsymbol{Y})$ is said to be an *augmented* version of $\boldsymbol{X}$. The auxiliary variable $\boldsymbol{Y}$ may be a natural part of a statistical model with hidden or unobserved data. However, in general $\boldsymbol{Y}$ is an artificially introduced variable that has no natural interpretation within the statistical model, and is used purely as a computational device. The combination of auxiliary variables with grouping and blocking yields some of the most celebrated algorithms.

■ **Example 1.8 (Slice Sampler for Probit Regression)** Suppose we wish to model the binary response data $\boldsymbol{y} = (y_1,\ldots,y_m)^\top$ with explanatory variables $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m$ using Bayesian probit regression [1]. The binary data is modelled as follows. Given the unknown parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ and the vector of observed covariates $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m$, each corresponding observation Y_i is assumed to follow a Bernoulli distribution with the success parameter

$$\mathbb{E}[Y_i \,|\, \boldsymbol{\beta}] = \Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta}), \quad i = 1,\ldots,m,$$

where Φ is the cdf of the standard normal density. Conditional on $\boldsymbol{\beta}$, the observations are assummed to be independent, so that the likelihood of the data is

$$\pi(\boldsymbol{y} \,|\, \boldsymbol{\beta}) = \prod_{i=1}^{m} [\Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta})]^{y_i}[1-\Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta})]^{1-y_i} = \prod_{i=1}^{m} \Phi((2y_i-1)\boldsymbol{x}_i^\top \boldsymbol{\beta}).$$

In the Bayesian paradigm we also assume that $\boldsymbol{\beta}$ follows a normal prior $\mathsf{N}(\mathbf{o},\varsigma^2\mathbf{I})$ with density $\varsigma^{-d}\varphi(\boldsymbol{\beta}/\varsigma)$. The objective now is to draw inference about the parameters via simulation from the posterior:

$$\pi(\boldsymbol{\beta} \,|\, \boldsymbol{y}) \propto \varphi(\boldsymbol{\beta}/\varsigma)\pi(\boldsymbol{y} \,|\, \boldsymbol{\beta})\,.$$

Sampling from the posterior directly could be difficult, and can be made easier through augmenting the observed data $\boldsymbol{y}$ with a variable $\boldsymbol{z}$, referred to as a latent variable. The idea [1] is to instead sample from the augmented posterior pdf $\pi(\boldsymbol{\beta},\boldsymbol{z}\,|\,\boldsymbol{y})$. To achieve this, note that since

$$\Phi((2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}) = \mathbb{E}\,\mathbb{I}\{Z\leq(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}\}$$

where $Z\sim\mathsf{N}(0,1)$, we can think of $\pi(\boldsymbol{y}\,|\,\boldsymbol{\beta})$ as the marginalization of the joint pdf

$$\pi(\boldsymbol{y},\boldsymbol{z}\,|\,\boldsymbol{\beta}) = \prod_i \varphi(z_i)\,\mathbb{I}\{z_i\leq(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}\},$$

because

$$\begin{aligned}\pi(\boldsymbol{y}\,|\,\boldsymbol{\beta}) &= \int_{\mathbb{R}^m}\prod_i \varphi(z_i)\,\mathbb{I}\{z_i\leq(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}\}\mathrm{d}\boldsymbol{z}\\ &= \prod_i\int_{\mathbb{R}}\varphi(z_i)\,\mathbb{I}\{z_i\leq(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}\}\mathrm{d}z_i\\ &= \prod_i \Phi((2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta}).\end{aligned}$$

Similarly, the posterior

$$\pi(\boldsymbol{\beta}\,|\,\boldsymbol{y}) \propto \varphi(\boldsymbol{\beta}/\varsigma)\prod_i\Phi((2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta})$$

can also be thought of as the marginal of the joint pdf

$$\pi(\boldsymbol{\beta},\boldsymbol{z}\,|\,\boldsymbol{y}) \propto \varphi(\boldsymbol{\beta}/\varsigma)\prod_i\varphi(z_i-(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta})\,\mathbb{I}\{z_i\geq 0\}.$$

The last can be written in vector form as

$$\pi(\boldsymbol{\beta},\boldsymbol{z}\,|\,\mathbf{X}) \propto \varphi(\boldsymbol{\beta}/\varsigma)\varphi(\boldsymbol{z}-\mathbf{X}\boldsymbol{\beta})\,\mathbb{I}\{\boldsymbol{z}\geq\mathbf{0}\},$$

where the matrix

$$\mathbf{X} := [(2y_1-1)\boldsymbol{x}_1,\ldots,(2y_m-1)\boldsymbol{x}_m]^\top.$$

The standard method of approximate simulation from the joint $\pi(\boldsymbol{\beta},\boldsymbol{z}\,|\,\mathbf{X})$ is via the Gibbs sampler [1]. The Gibbs sampler consists in iterative simulation from the conditional densities of $\boldsymbol{\beta}$ given $\boldsymbol{Z}$, and $\boldsymbol{Z}$ given $\boldsymbol{\beta}$. The second conditional pdf

$$\begin{aligned}\pi(\boldsymbol{z}\,|\,\mathbf{X},\boldsymbol{\beta}) &\propto \varphi(\boldsymbol{z}-\mathbf{X}\boldsymbol{\beta})\,\mathbb{I}\{\boldsymbol{z}\geq\mathbf{0}\}\\ &\propto \prod_{i=1}^m\varphi(z_i-(2y_i-1)\boldsymbol{x}_i^\top\boldsymbol{\beta})\,\mathbb{I}\{z_i\geq 0\}\end{aligned}$$

is a set of independent truncated normal distributions, and hence simple to simulate using rejection sampling. For the first conditional density, notice that the joint pdf of $\boldsymbol{\beta}$ and $\boldsymbol{z}$ can be rewritten as:

$$\begin{aligned}\pi(\boldsymbol{\beta},\boldsymbol{z}\,|\,\mathbf{X}) &\propto \exp\left(-\frac{1}{2}\|\boldsymbol{z}-\mathbf{X}\boldsymbol{\beta}\|^2-\frac{1}{2\varsigma^2}\|\boldsymbol{\beta}\|^2\right)\mathbb{I}\{\boldsymbol{z}\geq\mathbf{0}\}\\ &\propto \exp\left(-\frac{1}{2}\|\boldsymbol{z}\|^2+\boldsymbol{\beta}^\top\mathbf{X}^\top\boldsymbol{z}-\frac{1}{2}\boldsymbol{\beta}^\top\Sigma^{-1}\boldsymbol{\beta}\right)\mathbb{I}\{\boldsymbol{z}\geq\mathbf{0}\},\end{aligned}$$

where $\Sigma^{-1} := \mathbf{X}^\top\mathbf{X} + \varsigma^{-2}\mathbf{I}$. It follows from completing the square that

$$[\boldsymbol{\beta}\,|\,\boldsymbol{z}] \sim \mathsf{N}(\Sigma\mathbf{X}^\top\boldsymbol{z}, \Sigma),$$

that is, the conditional pdf $\pi(\boldsymbol{\beta}\,|\,\boldsymbol{z},\boldsymbol{y})$ is multivariate normal. Initializing from an arbitrary $\boldsymbol{\beta}$, the Gibbs sampling algorithm then involves iterative simulation of $[\boldsymbol{Z}\,|\,\boldsymbol{\beta}] \sim \mathsf{TN}_{(\mathbf{0},\infty)}(\mathbf{X}\boldsymbol{\beta}, \mathbf{I})$ followed by $[\boldsymbol{\beta}\,|\,\boldsymbol{Z}] \sim \mathsf{N}(\Sigma\mathbf{X}^\top\boldsymbol{Z}, \Sigma)$. In pseudo-code, we have the following *slice sampler*. The following MATLAB code

SLICE SAMPLER

Algorithm 1 Gibbs/Slice sampler for Probit Regression

```
Require: initial $\beta_0$; prior scale $\varsigma > 0$; data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^m$; chain length $t$
  $\mathbf{X} \leftarrow [(2y_1 - 1)\boldsymbol{x}_1, \ldots, (2y_m - 1)\boldsymbol{x}_m]^\top$
  $\Sigma \leftarrow \mathbf{X}^\top\mathbf{X} + \varsigma^{-2}\mathbf{I}$
  for $k = 1, \ldots, t$ do
      $[\boldsymbol{Z}\,|\,\beta_{k-1}] \sim \mathsf{TN}_{(\mathbf{0},\infty)}(\mathbf{X}\beta_{k-1}, \mathbf{I})$
      $[\beta_k\,|\,\boldsymbol{Z}] \sim \mathsf{N}(\Sigma\mathbf{X}^\top\boldsymbol{Z}, \Sigma)$
  return $\beta_1, \ldots, \beta_t$
```

shows an application of the slice sampling on the *Latent Membranous Lupus Nephritis* (or simply *lupus*) dataset described in [6, Table 1].

```
clear all, clc
load('lupus.csv'); % load data
Y=lupus(:,1);X=[ones(size(Y)),lupus(:,2:end)];%response and model matrix
[m, d] = size(X); % dimensions of problem
X=diag(2*Y-1)*X; % incorporate response into design matrix
varsig=sqrt(Inf); % prior scale parameter
Sig=inv(X'*X+eye(d)/varsig^2);L=chol(Sig,'lower');
b=[0,0,0]'; t=10^4;data=nan(t,3);
for k=1:t
    mu=X*b;
    Z=mu+trandn(-mu,Inf(size(mu)));
    b=Sig*X'*Z+L*randn(d,1);
    data(k,:)=b';
end
plot(data(:,2),data(:,3),'.','MarkerSize',.5)
```

The left pane of Figure 1.8 below shows the boxplots of the marginal distributions of the two *lupus* predictors and the intercept, estimated from $t = 10^4$ Gibbs simulations from the posterior with $\varsigma^2 = \infty$ (corresponding to a flat improper prior $\pi(\boldsymbol{\beta}) \propto 1$). The right pane shows the scatter plot of the coefficients, excluding the intercept.

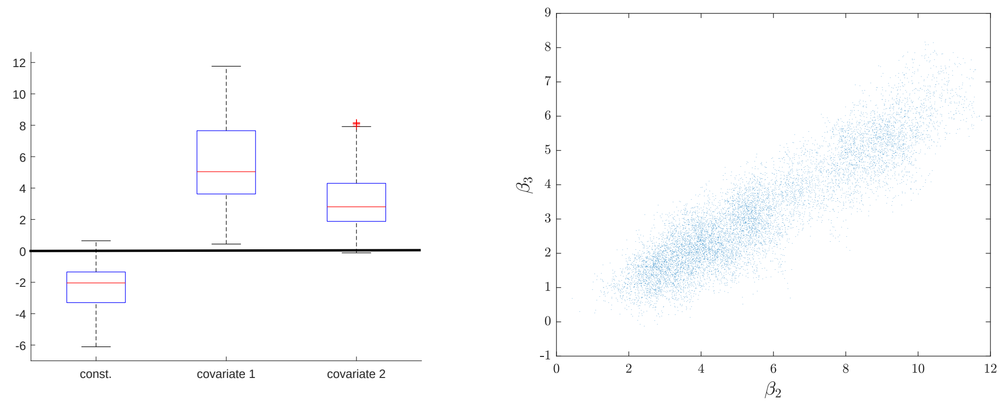

Figure 1.7: Left pane: boxplot of the marginal densities of $\beta_1, \beta_2, \beta_3$; right pane: scatterplot of the two predictor variables.

Unfortunately, the autocorrelations plot below does not look ideal, suggesting that this slice sampler does not converge very fast to the posterior density $\pi(\boldsymbol{\beta} \mid \mathbf{X})$.

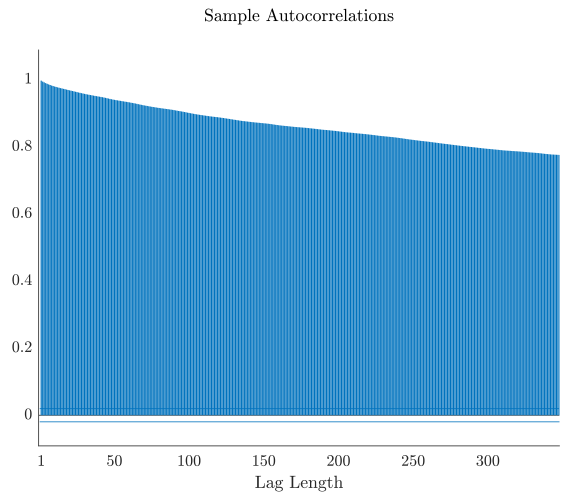

Figure 1.8: Autocorrelation plot of β_2 constructed from the output of the Gibbs sampler. The decay to zero seems to be very slow.

1.5 Approximate Simulation via Importance Sampling

Recall the accept-reject method for simulating from $\pi(\boldsymbol{x}) \propto f(\boldsymbol{x})$ (typically π is only known up to a normalizing constant) given that we have some enveloping proposal g such that $f(\boldsymbol{x}) \leq cg(\boldsymbol{x})$ for some constant $c > 0$.

Algorithm 2 Plain Accept-Reject

```
Require: proposal pdf $g$, enveloping constant $c$
  repeat
      Simulate $\boldsymbol{X} \sim g$ and $U \sim \mathsf{U}(0,1)$, independently.
      $W \leftarrow f(\boldsymbol{X})/(cg(\boldsymbol{X}))$
  until $U < W$
  return $\boldsymbol{X}$
```

What can we do if it is too hard to find c or if the ratio $\overset{f/g}{g/p}$ cannot be bounded? One answer is to use an approximate rejection method, which converges in a way similar to MCMC.

Algorithm 3 Importance Sampler

```
Require: proposal pdf $g$, number of steps/time $t$ (may be continuous)
  $N \leftarrow 0$
  repeat
      $N \leftarrow N + 1$
      Simulate $\boldsymbol{X}_N \sim g$
      $W_N \leftarrow f(\boldsymbol{X}_N)/g(\boldsymbol{X}_N)$
  until $W_1 + \cdots + W_N > t$
  return $\boldsymbol{X}_N$
```

Note that here we assume that f and g have the same support so that $W > 0$ always. If this is not the case then for a fixed k we will simulate from g until $W > 0$ is true.

Note that $N = N(t)$ is a function of t, because

$$N(t) = \min\{k : W_1 + \cdots + W_k > t\} .$$

Let $f(t) = o(1/t^p)$ be shorthand notation for

$$\lim_{t \uparrow \infty} t^p f(t) = 0 .$$

and $f(t) = \mathcal{O}(r^t)$ stand for:

$$\lim_{t \uparrow \infty} |f(t)/r^t| \leq \text{constant} < \infty .$$

How fast does the $\boldsymbol{X}_N$ converge to the target in terms of total variation distance as t becomes larger and larger? We have the following.

Useful Result 1.3: Rate of Convergence of Importance Sampler

If $\mathbb{E}_g W^{p+1} < \infty$ for some $p \geq 0$, then as $t \uparrow \infty$ the random variable $\boldsymbol{X}_{N(t)}$ converges at the polynomial speed

$$\sup_A |\mathbb{P}[\boldsymbol{X}_{N(t)} \in A] - \pi[A]| = o(1/t^p)\ .$$

Alternatively, if $\mathbb{E}\exp(\varepsilon W) < \infty$ for some $\varepsilon > 0$ (possibly very small), then $\boldsymbol{X}(t)$ converges at geometric speed, that is, there are constants $r \in (0,1)$ and $c_1 > 0$ such that

$$\sup_A |\mathbb{P}[\boldsymbol{X}_{N(t)} \in A] - \pi[A]| \leq c_1 r^t = \mathcal{O}(r^t)\ .$$

Compare this result with Theorem 1.1 and note how in both cases a bounded likelihood ratio W implies geometric convergence.

■ **Example 1.9 (Sampling from Gamma Density)** *As an example, suppose that* $\pi(x) = x^{\alpha-1}\exp(-x)/\Gamma(\alpha)$ *and* $g(x) = \beta\exp(-\beta x)$. *Then,* $w \propto x^{\alpha-1}\exp(-(1-\beta)x)$ *and hence we have*

$$\mathbb{E}[\exp(\varepsilon W)] = \mathbb{E}\exp(\varepsilon_1 X^{\alpha-1}\exp(-(1-\beta)X))$$

Since for any $\beta < 1$ *we have* $\sup_x\{x^{\alpha-1}\exp(-(1-\beta)x)\} < \infty$, *then* $\mathbb{E}[\exp(\varepsilon W)] < \infty$ *and the rate of convergence is geometric. For* $\beta = 1$ *we obtain that*

$$\mathbb{E}\exp(\varepsilon_1 X^{\alpha-1}) = \int_0^\infty \exp(-x + \varepsilon_1 x^{\alpha-1})\mathrm{d}x < \infty$$

provided that $\alpha \leq 2$.

For a polynomial rate of convergence, consider:

$$\begin{aligned}\mathbb{E}[W^k] &\propto \mathbb{E}[X^{k(\alpha-1)}\exp(-k(1-\beta)X)] \\ &\propto \int_0^\infty x^{k(\alpha-1)}\exp(-(k(1-\beta)+1)x)\mathrm{d}x,\end{aligned}$$

which is finite if $1 \leq \beta < 1 + \frac{1}{k}$. *For instance, if we select* $\beta = 1 + 1/10$, *then* $\mathbb{E}[W^k]$ *is finite for* $k \leq 9$. *Thus, the rate of convergence of the total variation distance is polynomial of order* $o(1/t^8)$. *On the other hand, if we select* $\beta \geq 1 + 1/k$, *then* $\boldsymbol{X}_{N(t)}$ *does not converge in distribution to the target.*

■ **Example 1.10 (Example 1.4 Continued.)** *In Example 1.4 we showed that when the support of the target* (1.7) *is bounded, then the likelihood ratio* $W(\boldsymbol{X}) < \infty$ *and thus the independence Metropolis-Hastings sampler is geometrically convergent. By the same argument, on a bounded domain the importance sampler in Algorithm 3 is geometrically convergent, because* $\mathbb{E}\exp(\varepsilon W) < \infty$.

What happens when we have an unbounded domain? We know that the independence sampler may fail to converge. Does the importance sampler in Algorithm 3 converge? Yes, but just barely, at the very *slow rate of $o(1/t)$. To see this note that due to symmetry*

$$
\begin{aligned}
-\frac{\sqrt{x_1^2+x_2^2}-|x_1|-|x_2|}{4} &= \frac{2|x_1x_2|/4}{\sqrt{x_1^2+x_2^2}+|x_1|+|x_2|} \\
&\leq \frac{x_1^2/2}{(\sqrt{2}+2)|x_1|} \qquad (x_1 = x_2) \\
&= \frac{|x_1|}{2(\sqrt{2}+2)}
\end{aligned}
$$

Hence, for the W in Algorithm 3 we have

$$
\begin{aligned}
\mathbb{E}W^2 &\leq \mathbb{E}\exp(|X_1|/(\sqrt{2}+2)) \\
&\leq \text{const.} \int_{\mathbb{R}} \exp\left(\frac{|x|}{\sqrt{2}+2} - \frac{|x|}{4}\right) \mathrm{d}x < \infty \,.
\end{aligned}
$$

Therefore, Result 1.3 with $p = 1$ implies that the convergence of the total variation error is of order $o(1/t)$.

■ **Example 1.11 (Geometric Convergence for Strongly Log-Concave Densities)** *Suppose that $\pi(\boldsymbol{x}) \propto \exp(-u(\boldsymbol{x}))$, where u is a strongly convex function, that is, there is a constant $m > 0$ such that*

$$u(\boldsymbol{x}) \geq u(\boldsymbol{y}) + (\boldsymbol{x}-\boldsymbol{y})^\top \frac{\partial u}{\partial \boldsymbol{y}} + \frac{m}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2, \quad \textit{for all } \boldsymbol{x},\boldsymbol{y} \tag{1.12}$$

STRONGLY LOG-CONCAVE

We then say that π is a strongly log-concave density. For many simple Bayesian models, the Bayesian likelihood functions are log-concave, which means that under a flat prior, the posterior π has a unique global maximum say $\boldsymbol{x}^*$ that satisfies $\frac{\partial u}{\partial \boldsymbol{x}}(\boldsymbol{x}^*) = \mathbf{0}$. Thus, convexity is useful for maximum likelihood inference (or finding the mode of the posterior under a flat prior).

The log-concavity also is useful for simulating from log-concave posteriors in Bayesian inference. Since posterior densities approach a normal density asymptotically, it makes sense to use a normal density $\mathsf{N}(\boldsymbol{\mu},\Sigma)$ as an importance sampling density in Algorithm 3. Then, the ratio $\pi(\boldsymbol{x})/g(\boldsymbol{x})$ is proportional to

$$w(\boldsymbol{x}) = \exp\left(-u(\boldsymbol{x}) + \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \,.$$

If we then select $\boldsymbol{\mu} = \boldsymbol{x}^*$ and use (1.12) with $\boldsymbol{y} = \boldsymbol{x}^*$, we obtain the bound:

$$w(\boldsymbol{x}) \leq \exp\left(-u(\boldsymbol{x}^*) - \frac{m}{2}\|\boldsymbol{x}-\boldsymbol{x}^*\|^2 + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x}^*)\right)$$

Note that if the likelihood with is bounded ($\sup_{\boldsymbol{x}} w(\boldsymbol{x}) < \infty$), then Algorithm 3 will converge at a geometrically fast speed, because $\mathbb{E}\exp(\varepsilon W(\boldsymbol{X})) < \infty$. The likelihood with be bounded provided that

$$(\boldsymbol{x}-\boldsymbol{x}^*)^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x}^*) \leq m\|\boldsymbol{x}-\boldsymbol{x}^*\|^2, \quad \textit{for all } \boldsymbol{x} \,.$$

The last is equivalent to requiring that

$$\sup_{\boldsymbol{y}\neq\mathbf{o}} \frac{\boldsymbol{y}^\top \Sigma^{-1}\boldsymbol{y}}{\|\boldsymbol{y}\|^2} \leq m,$$

that is, the largest eigen-value of Σ^{-1} must be $\leq m$. In other words, the smallest eigen-value of Σ must be $\geq m$. We can thus choose Σ to be proportional to the Hessian of $u(\boldsymbol{x})$ at $\boldsymbol{x}^*$ and then select the constant of proportionality in such a way that the smallest eigen-value of Σ is at least m. This will give us a very good proposal density p that will yield geometric convergence to the target π.

Alternatively, a simple heuristic is that if we make the variances in the covariance matrix Σ large and larger, then ultimately the importance sampler will be geometrically convergent. We can decide how large is large enough experimentally.

We can use all the $\boldsymbol{X}$'s simulated by Algorithm 3 to estimate quantities of interest such as $\alpha := \mathbb{E}_\pi \phi(\boldsymbol{X})$. Denote

$$T_k := W_1 + \cdots + W_k,$$

then our estimator of α is

$$\widehat{\alpha}(t) := \frac{\sum_{k=1}^N W_k \phi(\boldsymbol{X}_k)}{\sum_{k=1}^N W_k} = \frac{1}{T_N}\sum_{k=1}^N W_k \phi(\boldsymbol{X}_k)\,. \tag{1.13}$$

If we define $\mu := \mathbb{E}[W]$ and $Z_k := W_k(\phi(\boldsymbol{X}_k) - \alpha)$, then the error properties of the estimator are given in the following.

Useful Result 1.4: Error of Weighted Importance Sampling Estimator

Assuming that $\mathbb{E}[W^3]$ and $\mathbb{E}[Z^2]$ are finite, the squared bias of the estimator $\widehat{\alpha}(t)$ is bounded by:

$$(\mathbb{E}\widehat{\alpha}(t) - \alpha)^2 \leq \frac{\frac{4}{3}\mathbb{E}[W^3]\mathbb{E}[Z^2](\mu + \mathbb{E}[W^2]/t)}{(\mu t)^3} = \frac{4\mathbb{E}[W^3]\mathbb{E}[Z^2]}{3\mu^2 t^3} + o(1/t^3)$$

Assuming that $\mathbb{E}[W^2]$ and $\mathbb{E}[Z^2]$ are finite, the mean squared error of $\widehat{\alpha}(t)$ is bounded by:

$$\mathbb{E}(\widehat{\alpha}(t) - \alpha)^2 \leq \frac{\mathbb{E}Z^2}{t\mu} + \frac{\mathbb{E}[W^2]\mathbb{E}[Z^2]}{t^2\mu^2} = \frac{\mathbb{E}Z^2}{t\mu} + o(1/t)\,.$$

Note that all of the unknown quantities in the bound can be estimated using the natural estimators. This means that we can estimate these errors from the output of Algorithm 3. In particular,

$$\widehat{\mu} = \frac{1}{N}\sum_{k=1}^N W_k$$
$$\widehat{\mathbb{E}Z^2} = \frac{1}{N}\sum_{k=1}^N \{W_k(\phi(\boldsymbol{X}_k) - \widehat{\alpha}(t))\}^2\,.$$

To account for the computing cost of running any algorithm, it is not enough to examine the variance. Instead, we must consider the *work-normalized variance* (WNV), which is the variance of the estimator multiplied by the computational cost (running time). Running Algorithm 3 with input parameter t gives

WORK-NORMALIZED VARIANCE

us (on average) t/μ draws from g and evaluations of the ratio f/g. Hence, on average the WNV in this case is

$$\mathbb{V}(\widehat{\alpha}_t) \times \frac{t}{\mu} \leq \frac{\mathbb{E}Z^2}{\mu^2} + o(1)\,,$$

which accounts for both the accuracy and the computing cost of estimating α.

Example 1.12 (Comparison with vanilla Importance Sampling) *Suppose that* $\phi(\boldsymbol{x}) = \frac{g(\boldsymbol{x})}{f(\boldsymbol{x})} = W^{-1}(\boldsymbol{x})$ *so that* $\alpha = \frac{1}{\int f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}$ *is the reciprocal of the normalizing constant of the target* $\pi(\boldsymbol{x}) \propto f(\boldsymbol{x})$. *Note that* $\mu = \mathbb{E}W = 1/\alpha$ *is then the normalizing constant of* π. *The work-normalized variance of* $\widehat{\alpha}_t$ *is asymptotically*

$$\begin{aligned}\frac{\mathbb{E}Z^2}{\mu^2} &= \frac{\mathbb{E}(1-W\alpha)^2}{\mu^2} \\ &= \alpha^2\mathbb{E}(1-W\alpha)^2 = \alpha^2(\alpha^2\mathbb{E}W^2 - 1)\end{aligned}$$

Now, suppose that we use the vanilla importance sampling to estimate μ *via*

$$\widehat{\mu}_t = \frac{1}{t}\sum_{k=1}^{t} W(\boldsymbol{X}_k), \qquad \boldsymbol{X}_1,\ldots,\boldsymbol{X}_n \sim_{\text{iid}} g(\boldsymbol{x})$$

with variance $\mathbb{V}(\widehat{\mu}_t) = (\mathbb{E}W^2 - \mu^2)/t$ *and work-normalized variance* $\mathbb{V}(\widehat{\mu}_t) \times t = \mathbb{E}W^2 - \mu^2$.

Another estimator of α *is* $1/\widehat{\mu}_t$, *which by the Delta approximation has asymptotic variance given by*

$$\frac{\mathbb{E}W^2 - \mu^2}{t} \times \frac{1}{\mu^4} = \alpha^2(\alpha^2\mathbb{E}W^2 - 1)/t$$

In other words, the asymptotic work-normalized variance of $1/\widehat{\mu}_t$ *is the same as the asymptotic work-normalized variance of* $\widehat{\alpha}_t$.

Example 1.13 (Comparison with unnormalized Importance Sampling) *Consider estimating* $\alpha = \mathbb{E}_\pi \phi(\boldsymbol{X})$ *via the unnormalized importance sampling estimator:*

$$\tilde{\alpha}_t := \frac{\sum_{k=1}^t W_k\phi_k}{\sum_{k=1}^t W_k},$$

where $W_k := W(\boldsymbol{X}_k) = f(\boldsymbol{X}_k)/g(\boldsymbol{X}_k), \phi_k := \phi(\boldsymbol{X}_k)$, *and* $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n \sim_{\text{iid}} g(\boldsymbol{x})$. *Applying the Delta approximation on the ratio of averages estimator* $\frac{\frac{1}{t}\sum_{k=1}^t W_k\phi_k}{\frac{1}{t}\sum_{k=1}^t W_k}$ *yields the asymptotic work-normalized variance*

$$t\mathbb{V}(\tilde{\alpha}_t) \simeq \frac{\mathbb{V}(W\phi) - 2\alpha\mathbb{C}(W\phi, W) + \alpha^2\mathbb{V}(W)}{(\mathbb{E}W)^2} = \frac{\mathbb{E}Z^2}{\mu^2},$$

where $Z := W \times (\phi - \alpha)$. *Thus, the unnormalized importance sampling estimator* $\tilde{\alpha}_t$ *has (asymptotically) the same work-normalized variance as the estimator* $\widehat{\alpha}_t$ *in Theorem 3. This suggest that we should not expect much difference in the performance between the two types of importance sampling. Observe that* $\tilde{\alpha}_t$ *is an importance sampling estimator with a non-random computing effort (determined by* t*) and* $\widehat{\alpha}_t$ *is an importance sampling estimator with a random simulation effort (determined by the random variable* $N(t)$*).*

1.6 Thinning or Subsampling

We can rewrite Algorithm 3 in such a way that we do not store in memory anything but the final $\boldsymbol{X}_N$.

Algorithm 4 Importance Sampler (memory efficient)

```
Require: proposal pdf $g$, target pdf $\pi \propto f$, and time $t$
  $T \leftarrow 0$
  while $T < t$ do
      Simulate $\boldsymbol{X} \sim g$
      $T \leftarrow T + f(\boldsymbol{X})/g(\boldsymbol{X})$
  return $\boldsymbol{X}_N \leftarrow \boldsymbol{X}$
```

After running Algorithm 4 for a large time/steps t, we obtain a single $\boldsymbol{X}$ that is approximately distributed from the target pdf. What if we desire more than one sample, say m of them? A naive approach is to run Algorithm 4 independently m times. In this approach, however, all the $\boldsymbol{X}$'s will have the same accuracy in just the same way as the output of m independent Markov chains with the same initial condition.

Instead, just like with an MCMC sampler, we should run a single long chain and subsample it, that is, use thinning to obtain a sequence of (approximately) independent samples from the target. An Importance Sampler that uses thinning (or burn-in) of length b looks like this.

Algorithm 5 Importance Sampler with Thinning

```
Require: Running "time" $t$ and number of draws required $m$ (integer-valued)
  $W \leftarrow 0$
  $b \leftarrow t/m$, which is the thinning or burn-in length
  for $k = 1, \ldots, m$ do
      Draw $U \sim \mathsf{U}(0,1)$
      $T \leftarrow U \times W$
      while $T < b$ do
          Draw $\boldsymbol{X} \sim g$
          $W \leftarrow f(\boldsymbol{X})/g(\boldsymbol{X})$
          $T \leftarrow T + W$
      $\boldsymbol{X}_k \leftarrow \boldsymbol{X}$
  return $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$
```

Note that just like with MCMC that uses burn-in of length b, the quality of the sampled $\boldsymbol{X}$'s improves as b gets larger. Of course, a large b increases t (because $t = b \times m$ and we keep m fixed) and hence the computing cost.

Example 1.14 (Probit Regression Continued.) Consider again Example 1.8, where the target pdf for the lupus dataset is the posterior

$$\pi(\boldsymbol{\beta} \,|\, \mathbf{X}) \propto \prod_{i=1}^{55} \Phi((2y_i - 1)\boldsymbol{x}_i^\top \boldsymbol{\beta}) .$$

The following MATLAB code implements Algorithm 5 for approximate simulation from the log-concave pdf $\pi(\boldsymbol{\beta})$. The proposal density g is the pdf of $\mathsf{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ is the mode of $\pi(\boldsymbol{\beta})$ and $\Sigma = -4\mathbf{H}^{-1}$, where $\mathbf{H} := \frac{\partial^2 \pi}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top}$ is the Hessian matrix of π at $\boldsymbol{\mu}$. The likelihood ratio used was

$$W(\boldsymbol{\beta}) = \exp\left(6 + \tfrac{(\boldsymbol{\beta}-\boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{\beta}-\boldsymbol{\mu})}{2}\right) \prod_{i=1}^{55} \Phi((2y_i - 1)\boldsymbol{x}_i^\top \boldsymbol{\beta}) .$$

Note that here $\exp(6)$ is a scaling constant that ensures the mean $\mathbb{E}W$ is not too small.

```
clear all,clc
load('lupus.csv'); % load data
Y=lupus(:,1); X=[ones(size(Y)),lupus(:,2:end)];%response and model matrix
[ex, d] = size(X); % dimensions of problem
X=diag(2*Y-1)*X; % incorporate response into design matrix
% logarithm of joint density (up to a constant)
logf=@(b)sum(lnNpr(-Inf(ex,1),X*b));
% determine the mode using Newton Raphson
[mu,fval,exitflag,output,grad,H] = fminunc(@(x)(-logf(x)),zeros(d,1));
% scale parameter for proposal
Sigma=H\eye(d); B=chol(4*Sigma,'lower');
burn=1;m=10^4;    % initialization of alg
t=m*burn;  % algorithmic parameters
data=nan(m,d); W=0;
for k=1:m
    T=rand*W;
    while T<burn
        z=randn(d,1);
        b=mu+B*z;
        W=exp(logf(b)+0.5*norm(z)^2+6);
        T=T+W;
    end
    data(k,:)=b;
end
plot(data(:,2),data(:,3),'.','MarkerSize',.5)
```

This time the autocorrelation plot decays to zero much faster.

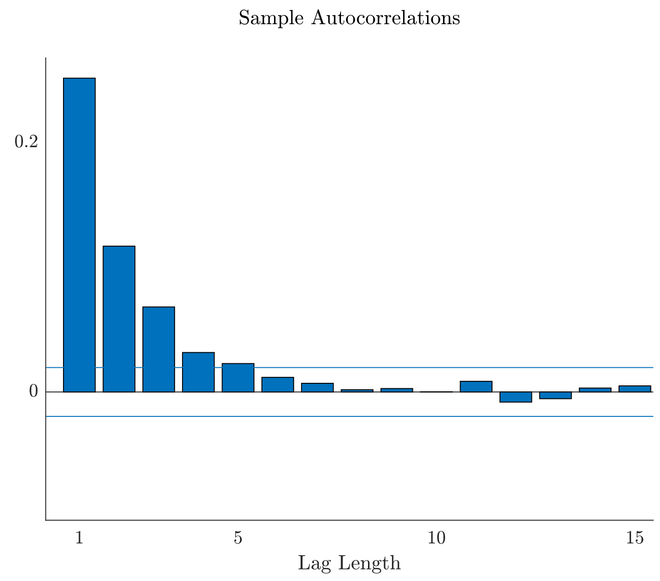

Figure 1.9: Autocorrelation plot of β_2 constructed from the output of Algorithm 5.

■ **Example 1.15 (Poisson Regression)** *In the Poisson regression we have count data $\boldsymbol{y}$ which, given the parameter $\boldsymbol{\beta}$ and the matrix with covariates $\mathbf{X}$, is modelled as*

$$(Y_i \mid \boldsymbol{\beta}, \boldsymbol{x}_i) \sim \mathsf{Poi}(\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}))$$

for all i. The Bayesian likelihood is then

$$\pi(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X}) = \exp\left(\sum_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} - \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}) - \ln(y_i!)\right) .$$

Therefore, using a flat improper prior $\pi(\boldsymbol{\beta}) \propto 1$ yields

$$\pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X}) \propto \exp\left(\sum_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} - \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})\right) .$$

As an example, consider the impact of research and development investment on the number of patents obtained by firms in a certain industry – the `RandDPatents.csv` dataset. For each of 14 companies we record the number of patents obtained in the last 3 years (this is y_i). We also record the company's research and development budget in thousands of dollars — this is the explanatory variable x_i in the Poisson model with mean: $\mathbb{E}[Y \mid \boldsymbol{x}_i, \boldsymbol{\beta}] = \exp(\beta_1 + x_i\beta_2)$.

The following MATLAB *code implements Algorithm 5 for approximate simulation from the posterior. The proposal density g is the pdf of $\mathsf{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ is the mode of $\pi(\boldsymbol{\beta})$ and $\Sigma = -2\mathbf{H}^{-1}$, where $\mathbf{H} := \frac{\partial^2 \pi}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top}$ is the Hessian matrix of the posterior at $\boldsymbol{\mu}$.*

```
clear all,clc
load('RandDPatents.csv'); % load data
% response and design matrix
Y = RandDPatents(:,1); X = [ones(size(Y)),RandDPatents(:,2)];
[ex, d] = size(X); % dimensions of problem
logf=@(b)(Y'*X*b-sum(exp(X*b))); % log-likelihood without constants
% determine the mode using Newton Raphson
[mu,fval,exitflag,output,grad,H] = fminunc(@(x)(-logf(x)),zeros(d,1));
% scale parameter for proposal
Sigma=H\eye(d); B=chol(2*Sigma,'lower');
burn=1;m=10^4;    % initialization of alg
t=m*burn;  % algorithmic parameters
data=nan(m,d); W=0; weight=nan(m,1);
for k=1:m
    T=rand*W;
    while T<burn
        z=randn(d,1);
        b=mu+B*z;
        W=exp(logf(b)+0.5*norm(z)^2-10);
        T=T+W;
    end
    data(k,:)=b; weight(k)=W;
end
plot(data(:,1),data(:,2),'.','MarkerSize',.5)
```

The following figure shows the scatterplot of $m = 10^4$ draws from the posterior. The posterior means were $\mathbb{E}[\boldsymbol{\beta}\,|\,\boldsymbol{y}] \approx (0.31, -0.012)^\top$. The scatter plot suggests that β_2 is significantly different from zero, that is, there is evidence that investment in research and development leads too more patents being filed.

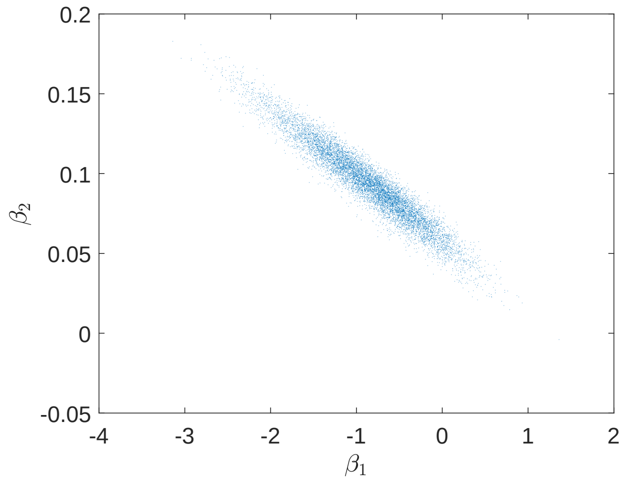

Figure 1.10: Scatterplot of β_1 versus β_2 draws from the posterior density.