MARKOV AND p–MARKOV PROCESSES

Recall the definition of a Markov process:

A stochastic process

$$X_0, X_1, \ldots, X_n, \ldots$$

is a Markov process if

$$\pi(x_{n+1} \mid x_0, x_1, \ldots, x_n) = \pi(x_{n+1} \mid x_n)$$

for all n.

Generalization: a stochastic process

$$X_0, X_1, \ldots, X_n, \ldots$$

is a p–Markov process if

$$\pi(x_{n+1} \mid x_0, x_1, \ldots, x_n) = \pi(x_{n+1} \mid \underbrace{x_{n-p+1}, \ldots, x_{n-1}, x_n}_{p})$$

for all n, where we interpret $X_j = 0$ for $j < 0$.

Markov = p–Markov with $p = 1$.

From p–Markov to Markov

Let $\{X_n\}$ be a p–Markov process.

Define

$$Z_n = \begin{bmatrix} X_n \\ X_{n-1} \\ \vdots \\ X_{n-p+1} \end{bmatrix}, \quad (X_{-j} = 0).$$

We have

$$\pi(z_{n+1} \mid z_n, z_{n-1}, \dots, z_0) = \pi(x_{n+1}, x_n, \dots, x_{n-p+2} \mid x_n, x_{n-1}, \dots, x_0).$$

Now we use a bit heuristics (everything can be done rigorously):

- If x_n is known, knowing $x_{n-1}, x_{n-2}, \ldots, x_0$ brings no extra information about x_n,
- If x_{n-1} is known, knowing $x_{n-2}, x_{n-3}, \ldots, x_0$ brings no extra information about x_{n-1},
- $\vdots$
- If x_{n-p+2} is known, knowing $x_{n-p+1}, x_{n-p}, \ldots, x_0$ brings no extra information about x_{n-p+2},

But since the process is p–Markov, knowing $x_{n-p}, x_{n-p-1}, \dots, x_0$ gives no extra information of x_{n+1}.

Conclusion: Knowing $x_{n-p}, x_{n-p-1}, \dots, x_0$ gives no information that would not be included in knowing $x_n, \dots, x_{n-p+2}$.

$$\pi(x_{n+1}, x_n, \dots, x_{n-p+2} \mid x_n, x_{n-1}, \dots x_{n-p+1}, \underbrace{x_{n-p}, \dots, x_0}_{\text{useless}})$$

$$= \pi(\underbrace{x_{n+1}, x_n, \dots, x_{n-p+2}}_{z_{n+1}} \mid \underbrace{x_n, x_{n-1}, \dots x_{n-p+1}}_{=z_n}),$$

in other words,

$$\pi(z_{n+1} \mid z_n, z_{n-1}, \dots, z_0) = \pi(z_{n+1} \mid z_n).$$

Moving window adaptation

Design a Metropolis-Hastings algorithm along the following guidelines:

- Random walk update,
- Adaptation: update the proposal distribution after every M steps,
- Proposal depends on few (two, say) previous blocks of length M.

Algorithm

1. Initialize $k = 0$, $C_k = \gamma^2 I$.
2. Generate a sample sequence of length M,

$$S_k = \left\{x_{kM+1}, x_{kM+2}, \ldots, x_{(k+1)M}\right\},$$

   using the random walk proposal

$$x_{\text{prop}} = x_{\text{curr}} + w, \quad w \sim \mathcal{N}(0, C_k)$$

3. Update

$$C_k \to C_{k+1} = \text{cov}(S_{k-1}, S_k) + \varepsilon I, \quad (S_{-1} = \emptyset).$$

4. Increase $k \to k+1$ and continue from 2 until desired sample size is reached.

Observe: The chain is not Markov, but it is $3M$–Markov.

We may write a proposal for z ($p = 3M$) as

$$z_{\text{prop}} = \begin{bmatrix} x_n + R_k^{\mathrm{T}} w \\ x_n \\ \vdots \\ x_{n-p+2} \end{bmatrix} \qquad (C_k = R_k^{\mathrm{T}} R_k)$$

$$= \begin{bmatrix} 1 & & & & \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-p+1} \end{bmatrix} + \begin{bmatrix} R_k^{\mathrm{T}} w \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$z_{\text{prop}} = V z_n + \eta,$$

where η depends on z_n, since the matrix C_k depends on x_j's with $j \geq n - 3M = n - p$, which are all included in z_n.

In other words: One step in z_n–history covers an x_n–history of length $3M$, which fully determines the updating matrix C_k.

Updating the covariance

Assume that $j = (k+1)M$:

$$x_0, \ldots, x_{(k-1)M}, \underbrace{x_{(k-1)M+1}, \ldots, x_{kM}}_{S_{k-1}}, \underbrace{x_{kM+1}, \ldots, x_{(k+1)M}}_{S_k}.$$

We have in memory

$$\overline{x}_{k-1} = \frac{1}{M} \sum_{j=(k-1)M+1}^{kM} x_j,$$

$$C_{k-1} = \frac{1}{M} \sum_{j=(k-1)M+1}^{kM} (x_j - \overline{x}_{k-1})(x_j - \overline{x}_{k-1})^{\mathrm{T}},$$

which have been computed when $j = kM$.

Calculate

$$\overline{x}_k = \frac{1}{M} \sum_{j=kM+1}^{(k+1)M} x_j,$$

$$C_k = \frac{1}{M} \sum_{j=kM+1}^{(k+1)M} (x_j - \overline{x}_k)(x_j - \overline{x}_k)^{\mathrm{T}},$$

Mean over $S_{k-1} \cup S_k$ is

$$\begin{aligned}
\overline{x} &= \frac{1}{2M} \sum_{j=(k-1)M+1}^{(k+1)M} x_j \\
&= \frac{1}{2} \left(\frac{1}{M} \sum_{j=(k-1)M+1}^{kM} + \frac{1}{M} \sum_{j=kM+1}^{(k+1)M} \right) x_j \\
&= \frac{1}{2} (\overline{x}_{k-1} + \overline{x}_k).
\end{aligned}$$

Write

$$
\begin{aligned}
C &= \frac{1}{2M} \sum_{j=(k-1)M+1}^{(k+1)M} (x_j - \overline{x})(x_j - \overline{x})^{\mathrm{T}} \\
&= \frac{1}{2} \left(\frac{1}{M} \sum_{j=(k-1)M+1}^{kM} + \frac{1}{M} \sum_{j=kM+1}^{(k+1)M} \right) (x_j - \overline{x})(x_j - \overline{x})^{\mathrm{T}}.
\end{aligned}
$$

The sums above are off-centered variances, and from the results of the previous lectures, we know that

$$
\frac{1}{M} \sum_{j=(k-1)M+1}^{kM} (x_j - \overline{x})(x_j - \overline{x})^{\mathrm{T}} = C_{k-1} + (\overline{x} - \overline{x}_{k-1})(\overline{x} - \overline{x}_{k-1})^{\mathrm{T}}.
$$

Similarly,

$$\frac{1}{M}\sum_{j=kM+1}^{(k+1)M}(x_j-\overline{x})(x_j-\overline{x})^{\mathrm{T}} = C_k + (\overline{x}-\overline{x}_k)(\overline{x}-\overline{x}_k)^{\mathrm{T}}.$$

Since

$$\overline{x}-\overline{x}_{k-1} = \frac{1}{2}(\overline{x}_k-\overline{x}_{k-1}) = -(\overline{x}-\overline{x}_k),$$

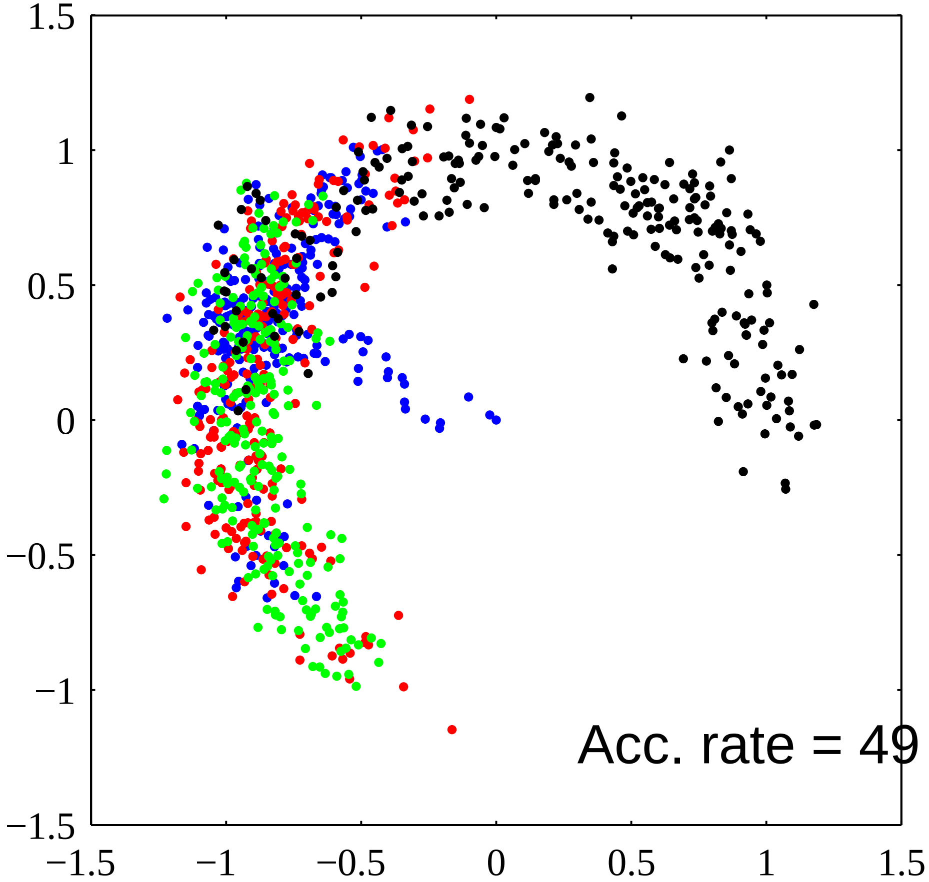
we obtain the updating formula,

$$C = \frac{1}{2}(C_{k-1}+C_k) + \frac{1}{4}(\overline{x}_k-\overline{x}_{k-1})(\overline{x}_k-\overline{x}_{k-1})^{\mathrm{T}}.$$

```
% Sampling with moving window adaptation

 SampleA = zeros(2,nsample);
 SampleA(:,1) = x0;
 x = x0;
 lpdf = -1/(2*sigr^2)*(norm(x)-r0)^2 - 1/(2*sigy^2)*(x(2)-1)^2;
 C2 = step^2*eye(2);
 x2 = zeros(2,1);
 mean = zeros(2,1);
 R = step*eye(2);
 accrate = 0;
 tempSample = [x];
 k = 0;
 S1 = [];
 S2 = [];
```

```
for j = 2:nsample

   % Draw the proposal
   xprop = x + R'*randn(2,1);
    lpdfprop =  -1/(2*sigr^2)*(norm(xprop)-r0)^2  ...
                 - 1/(2*sigy^2)*(xprop(2)-1)^2;
   % Check for acceptance
   if lpdfprop - lpdf >log(rand)
       %accept
       x = xprop;
       lpdf = lpdfprop;
       accrate = accrate + 1;
   end
   SampleA(:,j) = x;
   tempSample = [tempSample x];
```

```
    if mod(j,M) == 0
        % Update the proposal distribution
        S1 = S2;
        S2 = tempSample;
        tempSample = [];
        x1 = x2;
        C1 = C2;
        x2 = 1/M*sum(S2')';
        aux = S2 - x2*ones(1,M);
        C2 = 1/M*aux*aux';
        C = 1/2*(C1 + C2) + 1/4*(x1-x2)*(x1-x2)';
        R = chol(C);
        k = k+1;
    end

end rel_accrateA = 100*accrate/nsample;
```



Plotting: 1–500, 501–1000, 1001–1500, 1501–2000.

Observe: the sampler moves *along* the horseshoe, not across the gap, indicating that the step is *locally* adapted.