Multiple chain MCMC

The Metropolis-Hastings algorithm is a general purpose algorithm that is extremely useful in producing samples from arbitrary multivariate distributions. The principal problems it encounters in applications relate to

- **Choice of starting values:** Poorly chosen starting values can result in slow convergence; often deterministic numerical optimization methods can be used effectively to determine good starting values.
- **Posterior correlation:** High posterior correlation leads to inefficiency of Gibbs sampler moves, and slow convergence; this can sometimes be overcome by reparameterization.
- **Failure to explore the state space:** The MH algorithm can sometimes get trapped in localized regions of high probability density; this can sometimes be overcome by using *multiple chain* MCMC methods.

Multi-Mode Distributions

Example (Mixture of Bivariate Normals)

Suppose

$$\pi(\boldsymbol{x}) \equiv \omega \mathcal{N}_2(\boldsymbol{\mu}_1, \Sigma_1) + (1-\omega)\mathcal{N}_2(\boldsymbol{\mu}_2, \Sigma_2)$$

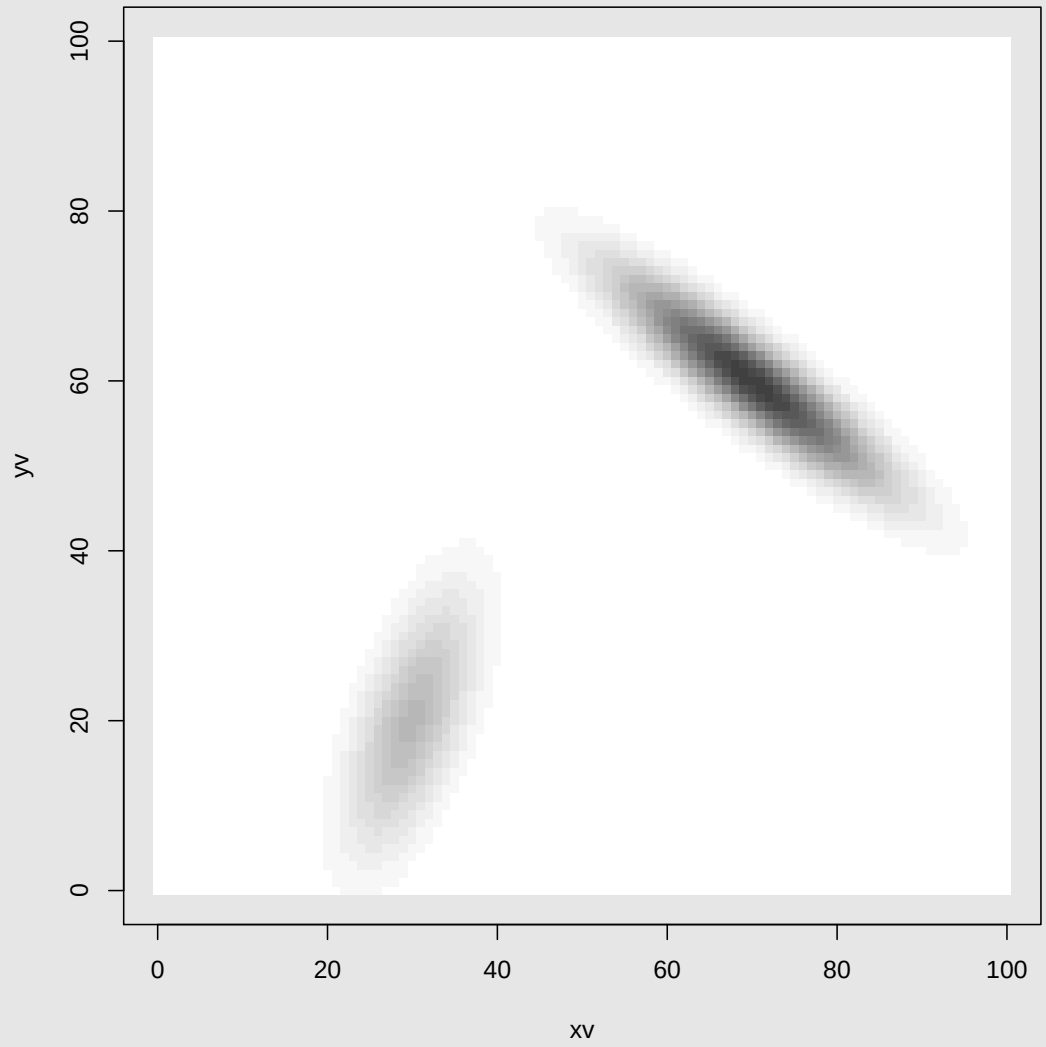
where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are relatively well separated (in light of the covariance matrices Σ_1 and Σ_2.

MCMC algorithms to sample from this bivariate density often encounter problems because they get stuck in the modes at $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, and cannot jump between them.

Multi-Mode Distributions

Example (Mixture of Bivariate Normals)

Mixture of Bivariate Normal densities.



Dominant-Mode Distribution

Example (The Witch's Hat)

In d dimensions, suppose

$$\pi(\boldsymbol{x}) = \omega \left(\frac{1}{\sigma}\right)^d \prod_{j=1}^{d} \phi\left(\frac{x_j - \theta_j}{\sigma}\right) + (1-\omega) \prod_{j=1}^{d} \mathbb{1}_{(0,1)}(x_j)$$

This density has a single mode at $\boldsymbol{\theta}$, but also a diffuse probability on the d-dimensional hypercube given by the second term.

MCMC algorithms can find it difficult to explore the "diffuse" region, because they get stuck in the mode, or *vice versa*.

Multiple chain MCMC

It is possible to use multiple Markov chain algorithms to produce variates from a target distribution π.

- **Replicate chains:** we run m chains with the **same** transition kernel P from **different starting values**, and then combine the chain as independent samples.
- **Parallel chains:** we run M chains each with its **own** transition kernel $P_1, \ldots, P_M$, such that the chain is reversible with respect to π: for $m = 1, \ldots, M$

$$\pi(x)P_m(x, y) = \pi(y)P_m(y, x)$$

and then the variates from the different chains are combined as independent samples.

In both of these cases, it is easy to see that the samples produced at stationarity are samples from π.

Multiple chain MCMC

- **Different chains:** we run M chains each with its **own** transition kernel $P_1, \ldots, P_M$, such that each chain is reversible with respect to **its own target** π_m where, say $\pi_1 \equiv \pi$. We then allow **exchange of information** between chains to facilitate sampling from π.

In this case it is not so straightforward to guarantee that the exchange of information does produce samples from π. This does need verification, once the exchange mechanism has been defined.

Such methods are called *multiple chain* or *population* MCMC.

Multiple chain MCMC

The simplest way to think about multiple chain methods in the case of different chains is via *auxiliary variables*, and an augmentation of the state space. For $m = 1, \ldots, M$, consider

- variable X_m
- (posterior) distribution $\pi_m(x)$
- irreducible transition kernel P_m, defined so that P_m and π_m exhibit detailed balance, and P_m defines a recurrent Markov chain

where, without loss of generality, π_1 is the distribution of interest, so that $X_2, \ldots, X_m$ are auxiliary variables.

Multiple chain MCMC

Consider the joint distribution for $\boldsymbol{X} = (X_1, \ldots, X_M)$, $\pi^M(x)$, defined by

$$\pi^M(\boldsymbol{x}) \propto \prod_{m=1}^{M} \pi_m(x_m).$$

By construction, the marginal distributions are $\pi_1, \ldots, \pi_M$.

Consider constructing a Markov chain on the extended state space, with π^M as the target distribution. The usual rules of Metropolis-Hastings apply - we propose a candidate new value, and accept it with some acceptance probability, otherwise we remain at the current value.

The values of x_1 that we collect across iterations are (dependent) samples from the marginal distribution, π_1.

Multiple chain MCMC: Types of move

1. **Mutation:** Select $m \in \{1, \ldots, M\}$, and perform a Metropolis-Hastings move using proposal density $q_m(x, z)$. If the current value for x_m is x, and the proposed value is z, accept the proposed point with probability

$$\alpha = \min \left\{ 1, \frac{\pi_m(z) q_m(z, x)}{\pi_m(x) q_m(x, z)} \right\}$$

   In the context of the vector $\boldsymbol{x}$, this move is essentially a Gibbs sampler move, as the full conditional for X_m is π_m.

2. **Exchange:** Select $l, m \in \{1, \ldots, M\}$, and propose the swap

$$x_l \longleftrightarrow x_m$$

   Accept this proposal with probability

$$\alpha = \min \left\{ 1, \frac{\pi_l(x_m) \pi_m(x_l)}{\pi_l(x_l) \pi_m(x_m)} \right\}.$$

Multiple chain MCMC: Types of move

In the case where all the π_m are equal to the target π, we can utilize the following move:

3. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, make the proposal z

$$z = x_m + u(x_l - x_m)$$

where u is a random variate sampled from the density

$$f(u) \propto \pi(x_m + u(x_l - x_m))$$

and then update x_l to z.

Multiple chain MCMC: Types of move

This move can be utilized in the case of **vector** state spaces. If each $\boldsymbol{x}$ is d-dimensional, then the move becomes

3*. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update $\boldsymbol{x}_l$, make the proposal $\boldsymbol{z}$

$$\boldsymbol{z} = \boldsymbol{x}_m + u(\boldsymbol{x}_l - \boldsymbol{x}_m)$$

where u is a scalar random variate sampled from the density

$$f(u) \propto |u|^{d-1} \pi(\boldsymbol{x}_m + u(\boldsymbol{x}_l - \boldsymbol{x}_m))$$

and then update $\boldsymbol{x}_l$ to $\boldsymbol{z}$.

Multiple chain MCMC: Types of move

A related move returns to basic ideas from optimization:

4. **Local optimum:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update $\boldsymbol{x}_l$, proceed as follows: starting at $\boldsymbol{x}_m$, use conjugate gradient or steepest descent to find the local mode of π, and denote this $\widehat{\boldsymbol{x}}_m$.

   Then make the proposal $\boldsymbol{z}$

   $$\boldsymbol{z} = \widehat{\boldsymbol{x}}_m + u(\boldsymbol{x}_l - \widehat{\boldsymbol{x}}_m)$$

   where u is a scalar random variate sampled from the density

   $$f(u) \propto |u|^{d-1} \pi(\widehat{\boldsymbol{x}}_m + u(\boldsymbol{x}_l - \widehat{\boldsymbol{x}}_m))$$

   and then update $\boldsymbol{x}_l$ to $\boldsymbol{z}$.

Multiple chain MCMC: Types of move

Again in the d-vector case, we have the final move type, inspired by ideas from genetic algorithms:

5. **Crossover:** Select $l, m \in \{1, \ldots, M\}$, and then select j uniformly from $\{1, \ldots, d-1\}$. Perform a crossover at position j, that is, attempt to replace $\boldsymbol{x}_l$ and $\boldsymbol{x}_m$ by

$$\boldsymbol{x}_l^{\text{new}} = (x_{l1}, \ldots, x_{lj}, x_{mj+1}, \ldots, x_{md})$$

$$\boldsymbol{x}_m^{\text{new}} = (x_{m1}, \ldots, x_{mj}, x_{lj+1}, \ldots, x_{ld})$$

that is, exchange the $(j+1, \ldots, d)$ portions of each vector.

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(\boldsymbol{x}_l^{\text{new}})\pi_m(\boldsymbol{x}_m^{\text{new}})}{\pi_l(\boldsymbol{x}_l)\pi_m(\boldsymbol{x}_m)}\right\}.$$

Parallel Tempering

Parallel Tempering is a multiple chain method that gives a specific construction of the distributions $\pi_1, \ldots, \pi_M$. Write the target density π as

$$\pi(x) \propto \exp\{-H(x)\}$$

and define

$$\pi_m(x) \propto \exp\left\{-\frac{H(x)}{T_m}\right\} = \pi(x)^{1/T_m}$$

where

$$1 = T_1 < T_2 < \cdots < T_M$$

represent a series of "temperatures". The densities $\pi_2, \ldots, \pi_M$ are *tempered* versions of π_1.

Parallel Tempering

Note that

$$\pi_m(x) = \frac{\exp\left\{-\frac{H(x)}{T_m}\right\}}{Z(T_m)}$$

where

$$Z(T_m) = \int \exp\left\{-\frac{H(x)}{T_m}\right\}\, dx$$

may not be available analytically.

This integral needs to be finite for π_m to be a proper distribution.

Parallel Tempering

Example (Two component Normal mixture)

For the two component mixture

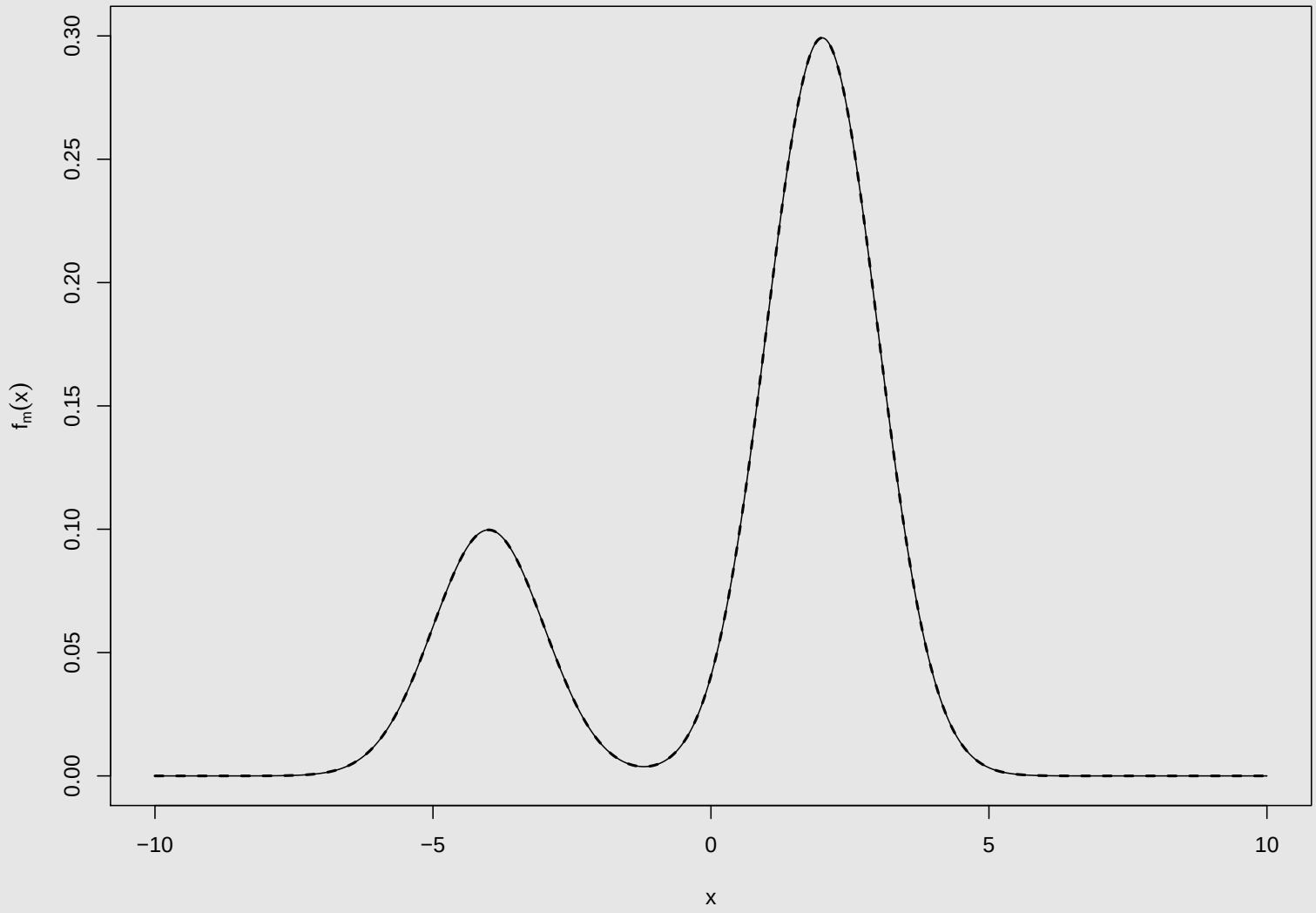
$$\pi(x) = \frac{1}{4}\phi(x+4) + \frac{3}{4}\phi(x-2)$$

the tempered distributions have modes that become increasingly flat.

Parallel Tempering

Example (Two component Normal mixture)

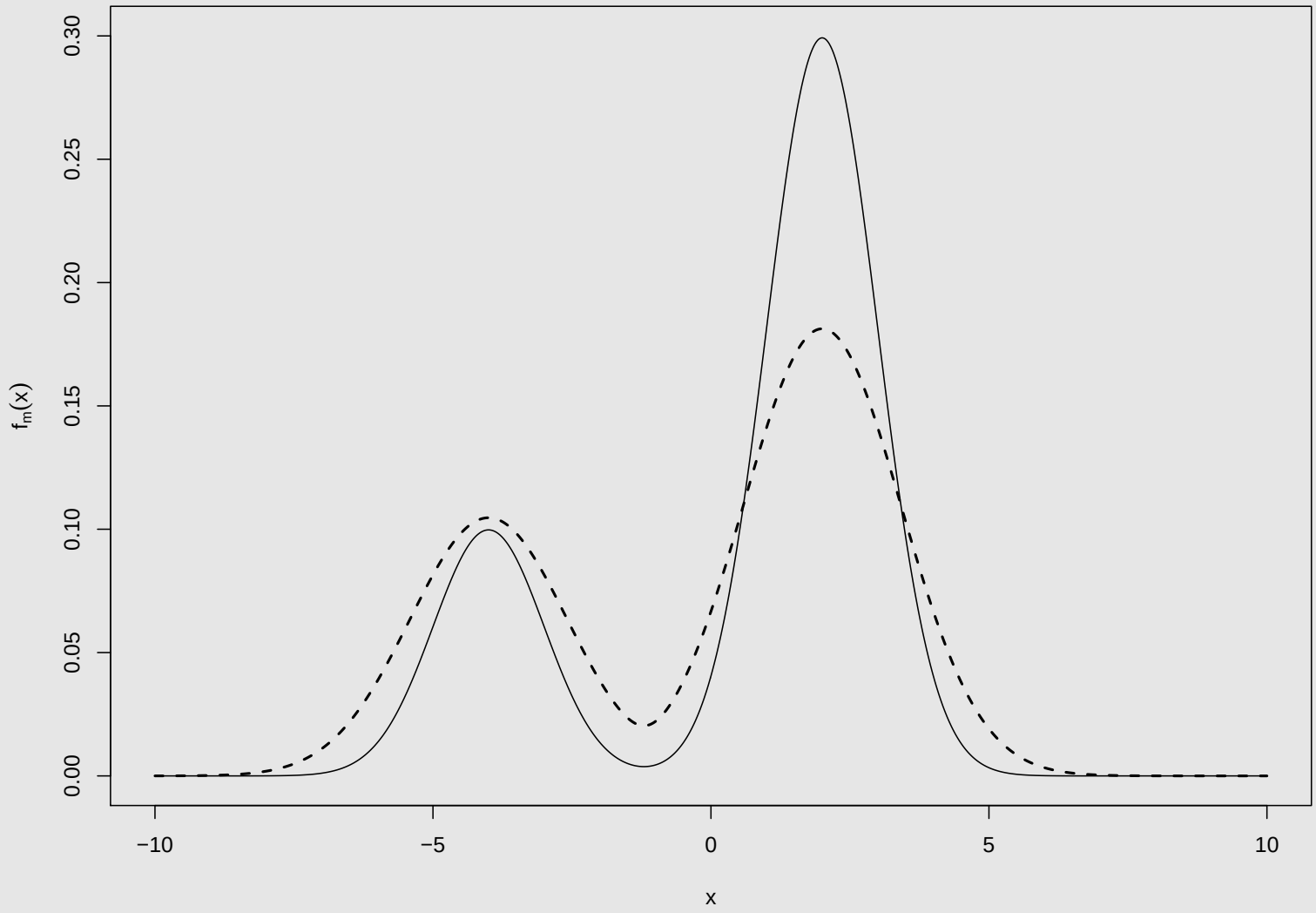
$T_1 = 1$:



Parallel Tempering

Example (Two component Normal mixture)

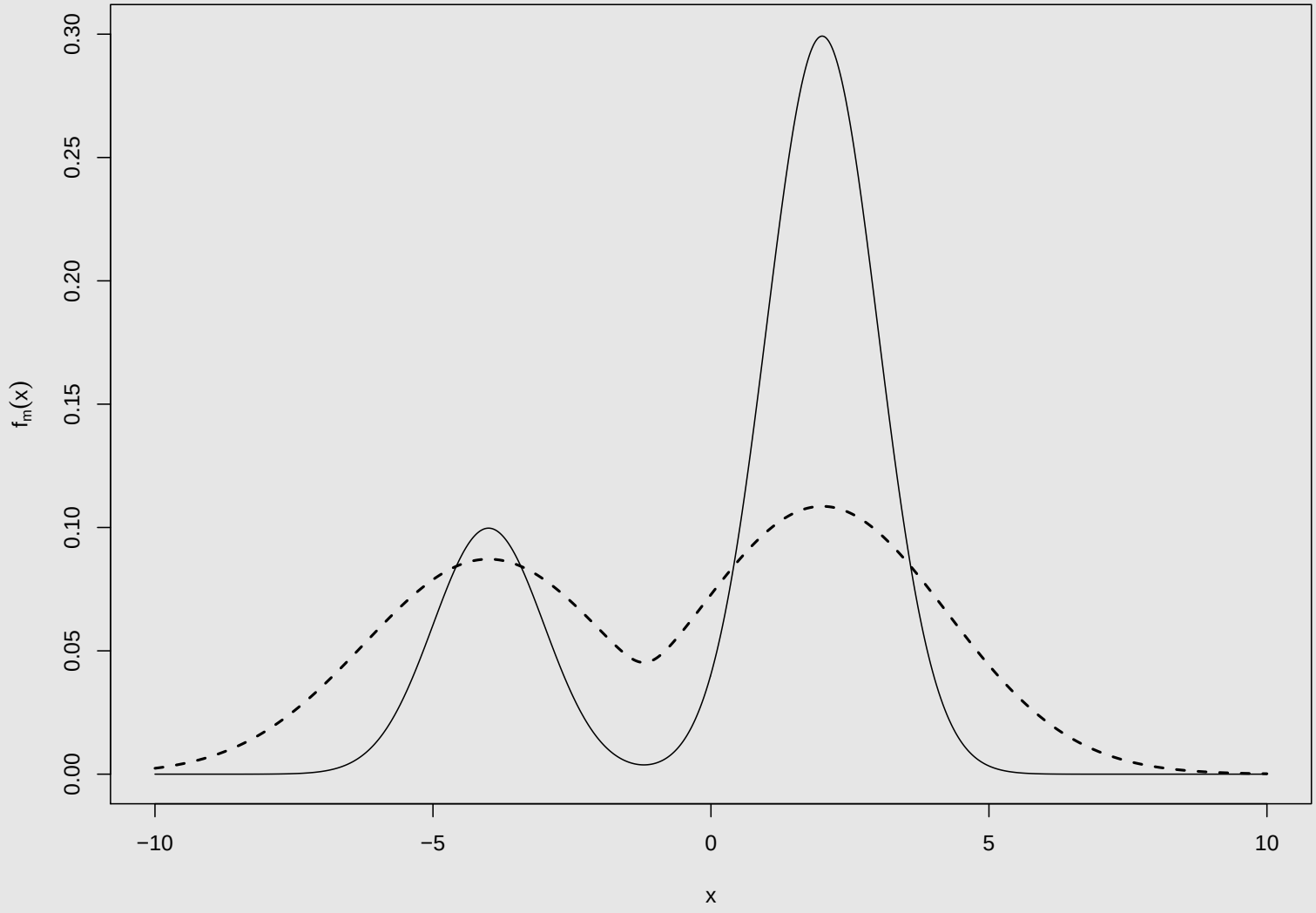
$T_2 = 2$:



Parallel Tempering

Example (Two component Normal mixture)

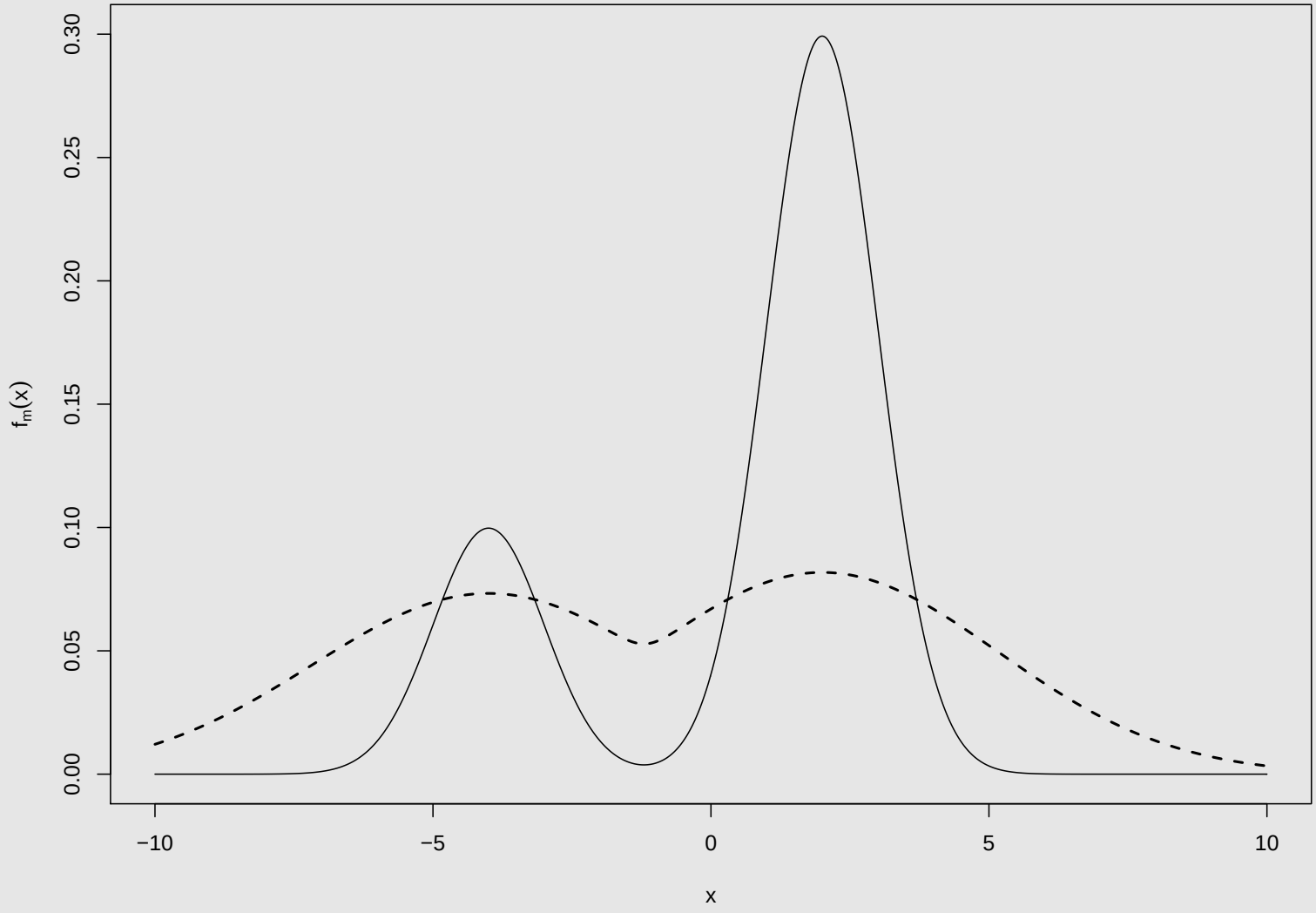
$T_3 = 5$:



Parallel Tempering

Example (Two component Normal mixture)

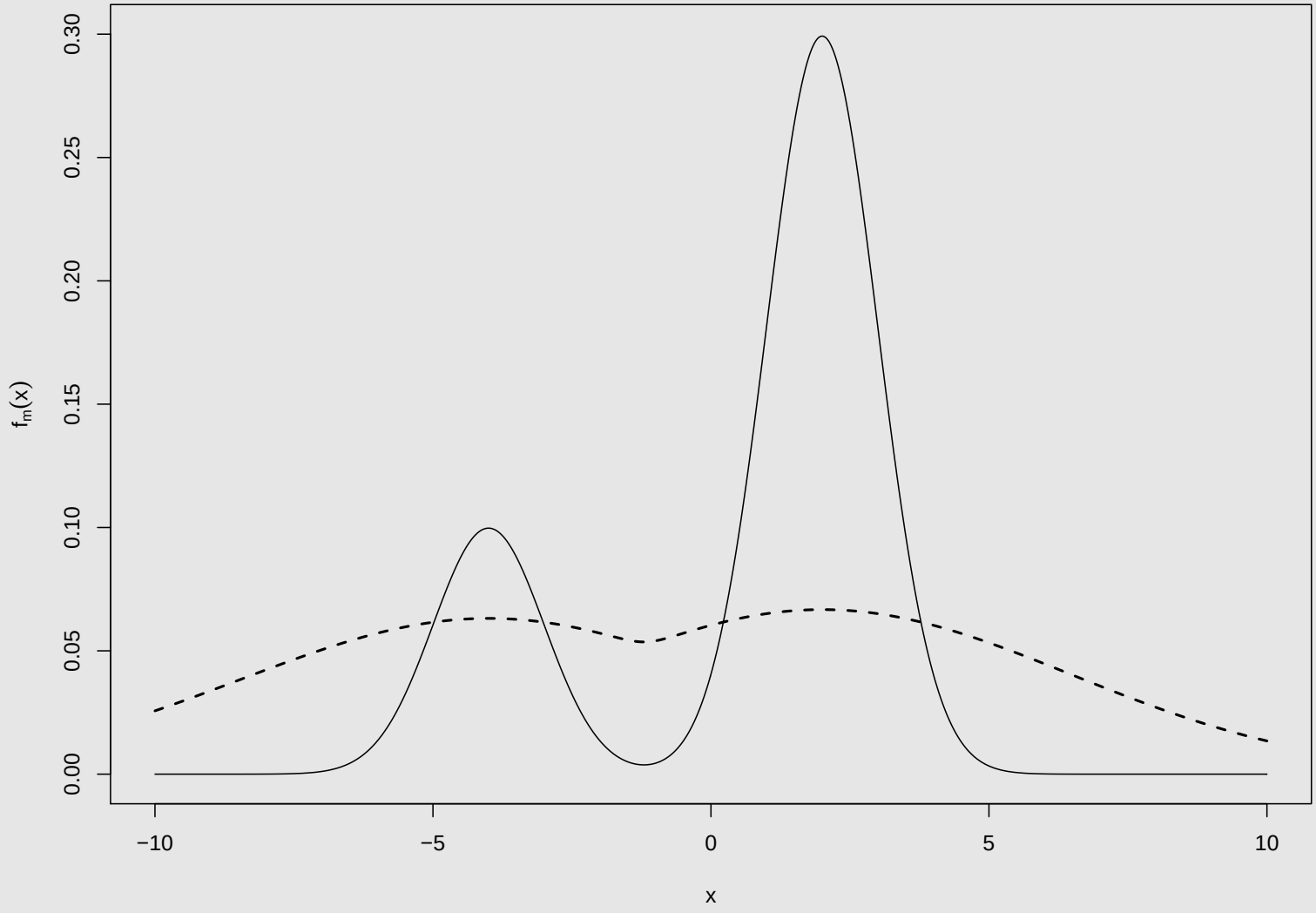
$T_4 = 10$:



Parallel Tempering

Example (Two component Normal mixture)

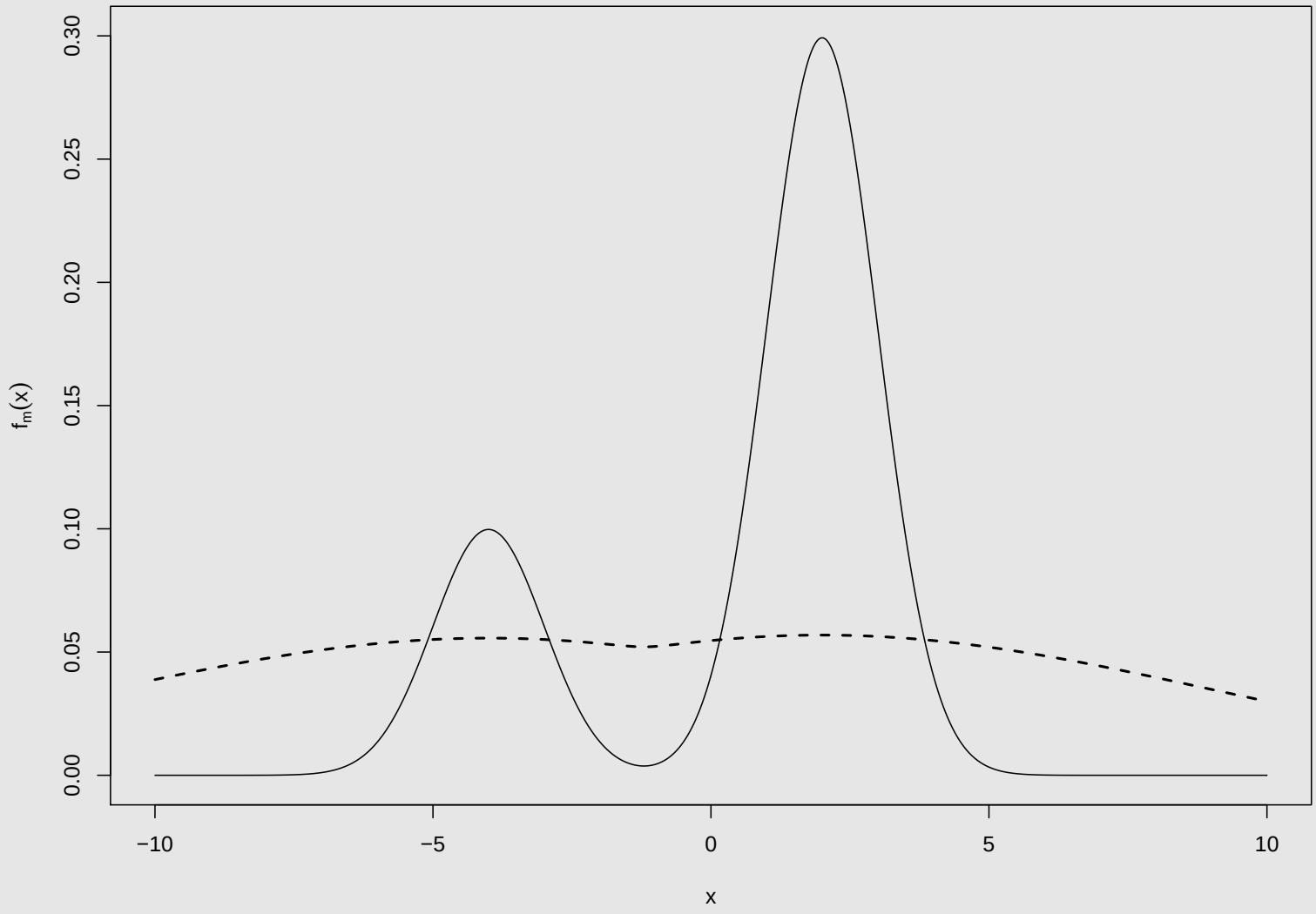
$T_5 = 20$:



Parallel Tempering

Example (Two component Normal mixture)

$T_6 = 50$:



Parallel Tempering

When T_m is large

$$\pi_m(x) \propto \exp\left\{-\frac{H(x)}{T_m}\right\}$$

is relatively "flat", that is, changes in x do not lead to large changes in $\pi_m(x)$.

Parallel tempering utilizes *within* chain and *between* chain updates. In the classical version, each iteration consists of

1. **Within:** an update for *each* chain using the Metropolis-Hastings kernels $P_1, \ldots, P_M$, then
2. **Between:** an update that attempts to swap x_m (from the chain for π_m at temperature T_m) with one of its temperature neighbours.

Parallel Tempering

Define the between-chain exchange probabilities

$$
\begin{aligned}
q_B(l,m) &= \mathbb{P}[\text{Swap chain } l \text{ with chain } m] \\
&= \begin{cases} \frac{1}{2} & 2 \leqslant l \leqslant M, m = l-1, l+1 \\ 1 & l = 1 \text{ and } m = 2, \text{ or } l = 1 \text{ and } m = 2 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Then the between-chain exchange that proposes to swap x_l with x_m is accepted with probability

$$
\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)q_B(m,l)}{\pi_l(x_l)\pi_m(x_m)q_B(l,m)}\right\}
$$

Parallel Tempering

For the tempered distributions

$$\frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)} = \exp\{-H(x_m)/T_l - H(x_l)/T_m + H(x_l)/T_l + H(x_m)/T_m\}$$

or equivalently

$$\exp\left\{(H(x_l) - H(x_m))\left(\frac{1}{T_l} - \frac{1}{T_m}\right)\right\}$$

The temperatures $T_2, \ldots, T_M$ should be chosen such that moves are accepted at an appropriate rate (neither too high nor too low).

The samples collected for x_1 across iterations represent a sample from the marginal $\pi_1 \equiv \pi$, the target distribution.

Parallel Tempering

Tempered moves can be proposed in sequence in an "up-down" fashion. An update for x_1 (in the target chain for $\pi_1 \equiv \pi$) can be proposed as follows.

For the "up phase":

Propose y_1 from MH kernel $q_1(x_1, y)$ in chain 1
Propose y_2 from MH kernel $q_2(y_1, y)$ in chain 2
Propose y_3 from MH kernel $q_3(y_2, y)$ in chain 3
...
Propose y_{M-1} from MH kernel $q_{M-1}(y_{M-2}, y)$ in chain $M-1$
Propose y_M from MH kernel $q_M(y_{M-1}, y)$ in chain M

Parallel Tempering

For the "down phase":

Propose z_M from MH kernel $q_M(y_M, z)$ in chain M
Propose z_{M-1} from MH kernel $q_{M-1}(z_M, z)$ in chain M-1
Propose z_{M-2} from MH kernel $q_{M-2}(z_{M-1}, z)$ in chain M-2
...
Propose z_2 from MH kernel $q_2(z_3, z)$ in chain 2
Propose z_1 from MH kernel $q_1(z_2, z)$ in chain 1

This up and down sequence has proposed a move in the π_1 chain $x_1 \longrightarrow z_1$ via the intermediate steps through the other tempered chains.

Parallel Tempering

The acceptance probability for the move is the minimum of 1 and

$$\frac{\pi_1(z_1)}{\pi_1(x_1)} \frac{q_{\overline{U}}(z_1, \ldots, z_M, y_M) q_{\overline{D}}(y_M, \ldots, y_1, x_1)}{q_{\underline{U}}(x_1, y_1, \ldots, y_M) q_{\underline{D}}(y_M, z_M, \ldots, z_1)}$$

where

$$\begin{aligned}
q_{\overline{U}}(z_1, \ldots, z_M, y_M) &= q_1(z_1, z_2) q_2(z_2, z_3) \cdots q_M(z_M, y_M) \\
q_{\overline{D}}(y_M, \ldots, y_1, x_1) &= q_M(y_M, y_{M-1}) q_{M-1}(y_{M-1}, y_{M-2}) \cdots q_1(y_1, x_1) \\
q_{\underline{U}}(x_1, y_1, \ldots, y_M) &= q_1(x_1, y_1) q_2(y_1, y_2) \cdots q_M(y_{M-1}, y_M) \\
q_{\underline{D}}(y_M, z_M, \ldots, z_1) &= q_M(y_M, z_M) q_{M-1}(z_M, z_{M-1}) \cdots q_1(z_2, z_1)
\end{aligned}$$

If all of the proposals are symmetric $q_m(x, y) = q_m(y, x)$, then the ratio simplifies to

$$\frac{\pi_1(z_1)}{\pi_1(x_1)}$$

Evolutionary Monte Carlo

Evolutionary Monte Carlo combines the move types outlined above (mutation, exchange, crossover, snooker) with tempered distributions in order to make exploration of the state space more effective.

The Metropolis-Hastings acceptance probability is reasonably straightforward to evaluate for each type of move. For a crossover move, a common approach to choosing the two "parent" vectors that participate in the crossover is to examine the current configurations, and to select the first parent to be $\boldsymbol{x}_l$ with probability p_l proportional to

$$\exp\{-\beta H(\boldsymbol{x}_l)\}$$

where $\beta > 0$, and the second parent uniformly from the remaining population.

Evolutionary Monte Carlo

The acceptance probability is computed by considering the various probabilities and densities in the proposal steps. For the crossover move

- Pick parent 1, $\boldsymbol{x}_l$, with probability p_l
- Pick parent 2, $\boldsymbol{x}_m$, uniformly, with probability $1/(M-1)$
- Pick the location of the crossover j, uniformly, with probability $1/(d-1)$
- Perform the crossover to obtain $\boldsymbol{x}_l^{\text{new}}$ and $\boldsymbol{x}_m^{\text{new}}$.

This yields the **numerator** in the Hastings ratio

$$p_m \times \frac{1}{M-1} \times \frac{1}{d-1} \times \pi_l(\boldsymbol{x}_l^{\text{new}}) \times \pi_m(\boldsymbol{x}_m^{\text{new}})$$

Considering the reverse move yields the denominator.

Simulated Tempering

In *simulated tempering*, the temperatures are also treated as parameters quantities that are updated during the MCMC updates. Suppose

$$\pi_m(x, \tau_m) \propto \frac{\exp\{-H(x)/\tau_m\}}{Z(\tau_m)}$$

where $Z(\tau_m)$ is the normalizing constant for the mth tempered (conditional) distribution. A Metropolis-Hastings algorithm is constructed on the augmented state space

$$(x_1, \tau_1), (x_2, \tau_2), \ldots, (x_M, \tau_M)$$

and the sampled collected from chain 1 are samples from the target $\pi_1 \equiv \pi$.

Simulated Tempering

Mutation and exchange moves can be implemented. For the **mutation** move, chain m is selected uniformly from collection of chains, and then the pair (x_m, τ_m) is updated using a transition proposal

$$q_m((x_m, \tau_m), (x_m^{\text{new}}, \tau_m^{\text{new}}))$$

with the usual acceptance probability.

For the **exchange** move between l and m, the proposed exchange is accepted with probability

$$\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)}\right\}$$

where the probabilities of choosing pairs of chains l and m to attempt the exchange are equal.

Simulated Tempering

A Gibbs sampler update, involving proposals for x, and then for τ_l, can also be carried out. Specifically, we may propose an exchange for τ_l whilst holding x_l constant.

If l is selected uniformly from $\{1, \ldots, M\}$, then another index m is selected with probability q_{lm}, and a temperature swap $\tau_l \longleftrightarrow \tau_m$ is proposed. In this case, the acceptance probability is

$$\alpha = \min\left\{1, \frac{\pi_m(x_l)}{\pi_l(x_l)} \frac{q_{ml}}{q_{lm}}\right\}$$

Here

$$\frac{\pi_m(x_l)}{\pi_l(x_l)} = \frac{Z(\tau_m)}{Z(\tau_l)} \exp\left\{-H(x_l)\left(\frac{1}{\tau_m} - \frac{1}{\tau_l}\right)\right\}$$

In this formula, the normalizing constant ratio

$$\frac{Z(\tau_m)}{Z(\tau_l)}$$

must be computed. Here

$$Z(\tau) = \int \exp\{-H(x)/\tau\}\, dx$$

which, in the multivariate case is a d-dimensional integral that needs to be finite for each possible τ.

In some cases, analytical expressions can be obtained, but more generally an estimate $\widehat{Z}(\tau)$ must be used.

Simulated Annealing

Simulated annealing is a related optimization technique that utilizes *MCMC* and a temperature ladder. Suppose that the function $H(x)$ is to be minimized. Define the tempered probability distribution

$$\pi_T(x) \propto \exp\{-H(x)/T\}$$

and consider the sequence of **decreasing** temperatures

$$T_1 > T_2 > \cdots$$

Simulated annealing proceeds by running MCMC at the highest temperature T_1 for a fixed number N_1 of iterations, then changing temperatures to T_2 and running MCMC for N_2 iterations, and so on.

Simulated Annealing

As $i \longrightarrow \infty$ with $T_i \longrightarrow 0$, the MCMC samples become focussed at the mode(s) of $\pi(x)$, that is, the minima of $H(x)$.
Theoretically, the logarithmic rate, where

$$T_i = \frac{K}{\log(M_i)} \qquad \text{with} \qquad M_i = \sum_{j=1}^{i} N_i$$

guarantees convergence of the algorithm to the global maximum of π.

This rate is extremely slow, and typically geometric or linear cooling schedules are used; they generally have reasonable performance.

Simulated annealing is available as an option in the `optim` function in R.