Metropolis–Hastings Algorithm

Apurva Nakade

2025-03-24

Table of contents

Random Walks Metropolis–Hastings Algorithm 1
    Proposal Distribution 2
    Choosing the Proposal Distribution 5
Metropolis–Hastings Algorithm 6
    Structure of Metropolis–Hastings Algorithm 6
    Proposal Distribution 7
    Acceptance Criterion 7
Symmetric Proposal Distribution 8
Metropolis Markov Chain 9
    Detailed Balance Equations 10

Metropolis–Hastings algorithm is a Markov Chain Monte Carlo (MCMC) method used to sample from a probability distribution. It is a type of a rejection sampling algorithm where we generate a sequence of samples from a target distribution by proposing a new sample and accepting or rejecting it based on a certain criterion. The algorithm is widely used in Bayesian statistics, statistical physics, and machine learning.

We'll start a few examples to understand the algorithm and then discuss the general algorithm.

Random Walks Metropolis–Hastings Algorithm

Consider a region Ω in $\mathbb{R}^n$. Suppose we want to sample uniformly from this region. One way to do this is through rejection sampling. We envelope the region in simple shape like a cube and sample uniformly from the cube. If the sample lies in the region Ω, we accept it, otherwise we reject it. The efficiency of this method depends on the ratio of the volume of the cube to the volume of the region Ω. If the region is highly non-convex, this ratio can be very small and the rejection sampling can be very inefficient.

An alternative method is to use random walks. We start at a point x_0 in the region Ω and take a random step in a random direction. If the new point x_1 is in the region Ω, we accept it, otherwise we stay at the point x_0. We repeat this process to generate a sequence of points. This method is more efficient than rejection sampling for highly non-convex regions.

The rationale behind this is that if a point is in the region Ω, then a point near it is also likely to be in the region. We can use this idea to sample from a probability distribution. The random walk Metropolis–Hastings algorithm for generating N samples from a region Ω is as follows:

```
1. Start at a point $x_0$ in the region $\Omega$.
2. For $i = 0, 1, \dots, N-1$:
    1. Generate a random step $\sigma x$.
    2. Compute the new point $y$ **near** $x_i$.
    3. If $y$ is in the region $\Omega$,
        1. Set $x_{i+1} = y$.
    4. Else,
        1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \dots, x_{N-1}$.
```

This is an example of a Metropolis–Hastings algorithm. The acceptance criterion is that the *new point y should be in the region Ω*.

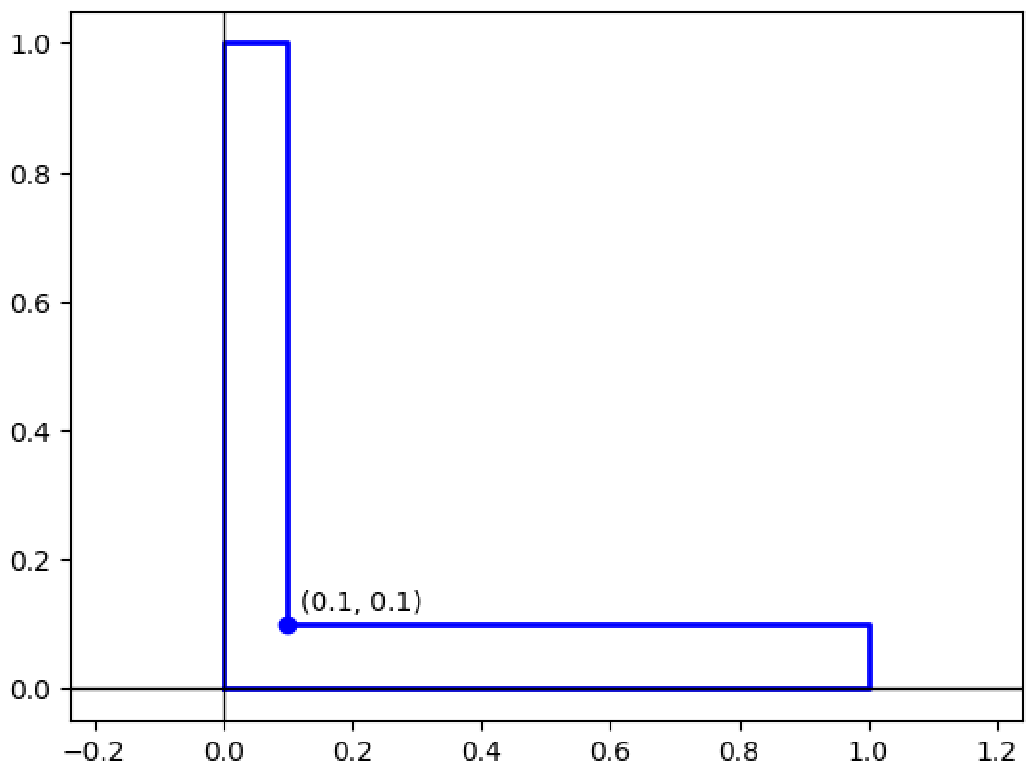
Proposal Distribution

The missing piece in the above algorithm is the method to generate a new point y near the current point x_i. This is done using a **proposal distribution**. The proposal distribution can be almost of any form, but it should be easy to sample from. The choice of the proposal distribution is crucial for the efficiency of the algorithm. A good proposal distribution should be able to explore the region Ω efficiently.

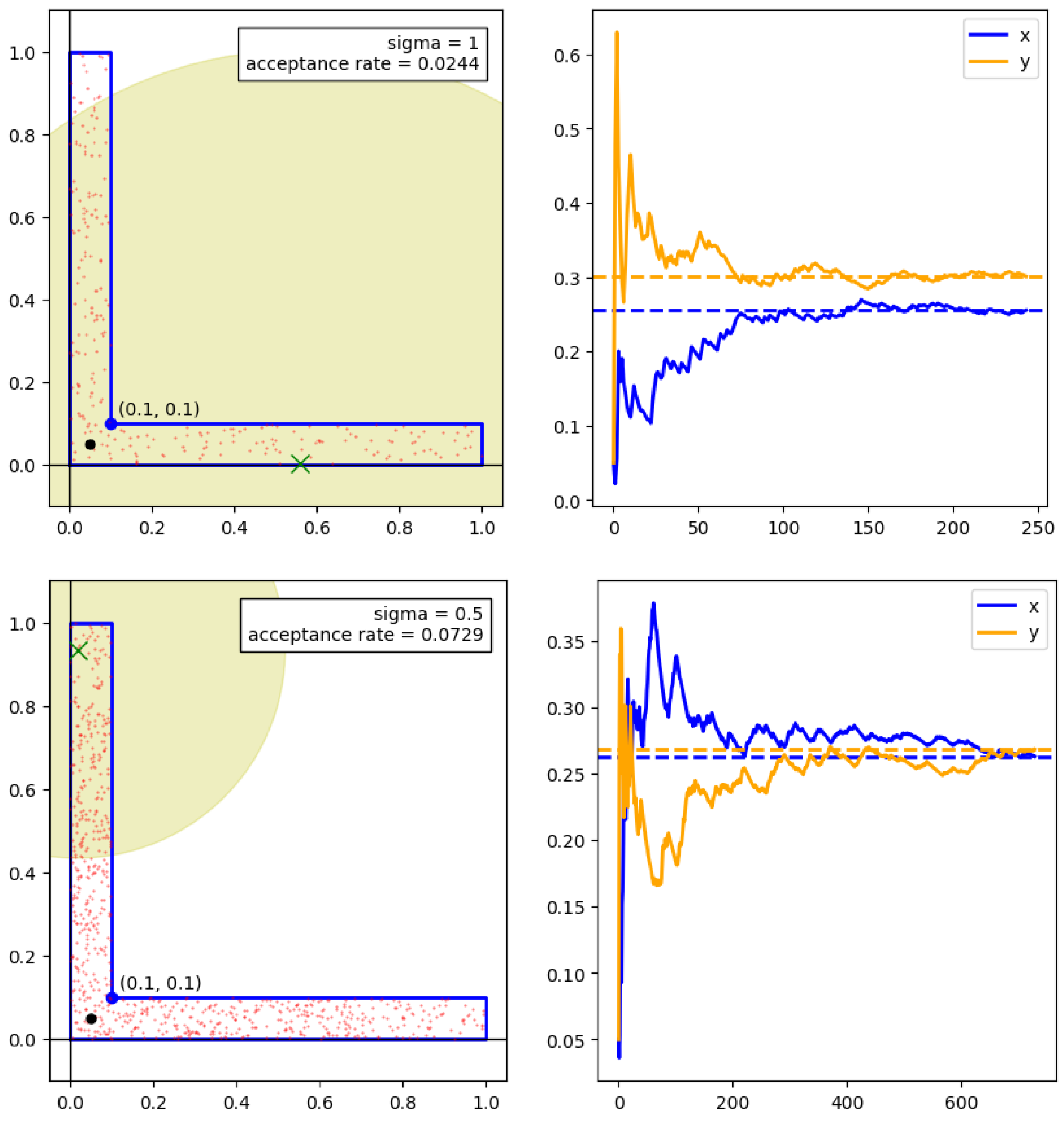
In the case of random walks, there are two common choices for the proposal distribution:

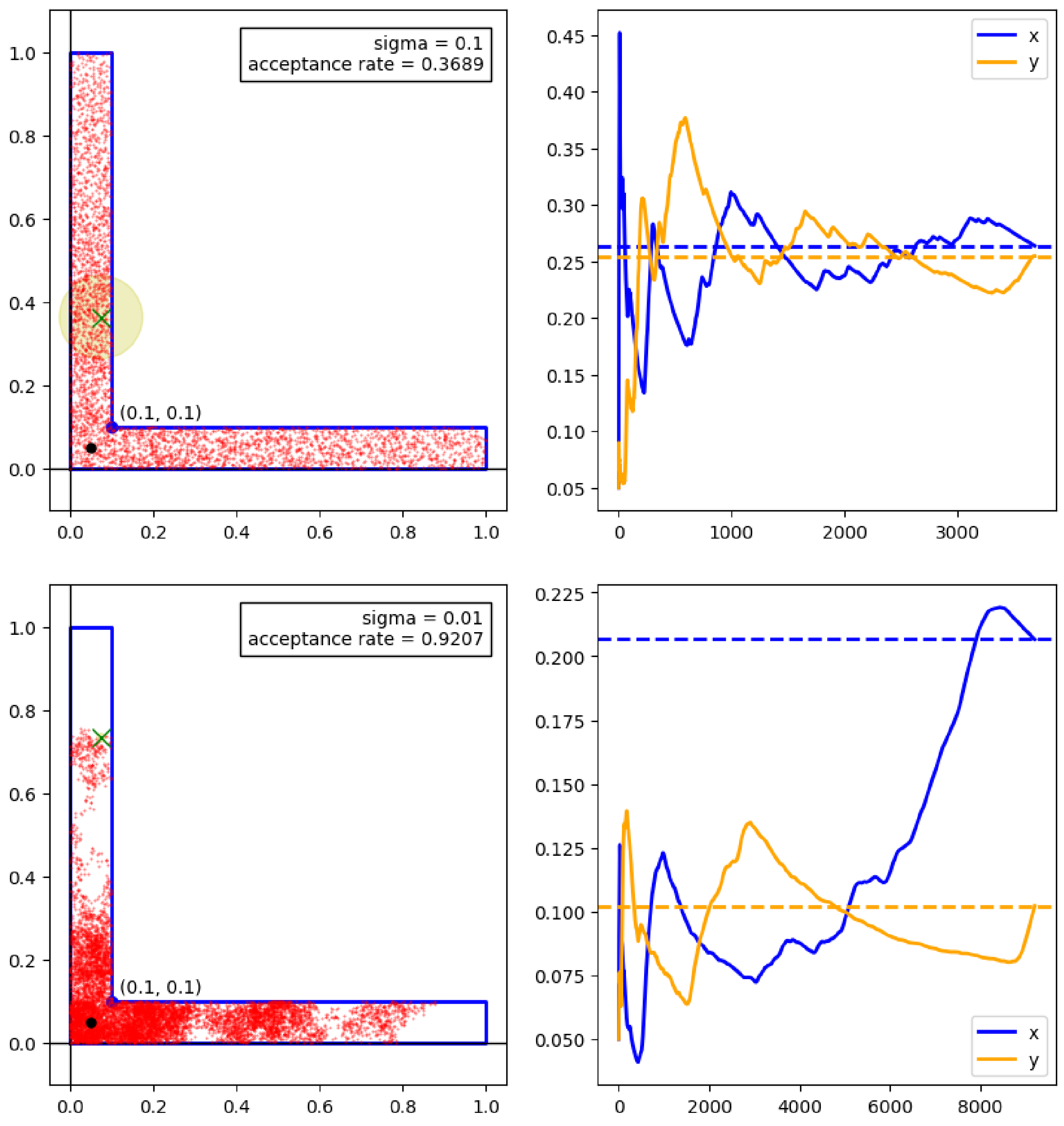
1. **Gaussian proposal**: We sample a new point from a Gaussian distribution centered at the current point x_i with a certain variance.
2. **Uniform proposal**: We sample a new point from a uniform distribution in a neighborhood of the current point x_i.

Example 0.1. Consider the L-shaped region in $\mathbb{R}^2$ as shown below. This is a non-convex region and rejection sampling would only provide an efficiency of $0.1 + 0.1 - 0.01 = 0.18$. We'll use the Metropolis–Hastings algorithm to sample from this region with higher efficiency.



The images below show scatter plots for the samples generated using the Metropolis–Hastings algorithm with four different choices of the Gaussian proposal distribution with standard deviations 1, 0.5, 0.2, and 0.01. The plots to the right are the running averages of the x and y coordinates. The scatter plot and the running averages provide a visual representation of the convergence of the samples to the target distribution.


sigma = 1
acceptance rate = 0.0244
(0.1, 0.1)
x
y
sigma = 0.5
acceptance rate = 0.0729
(0.1, 0.1)
x
y



Choosing the Proposal Distribution

Notice that there is a tradeoff between the acceptance rate and the rate of convergence. In the above example,

- The proposal distribution with $\sigma = 1$ has the lowest acceptance rate but the samples are well spread out and the underlying markov chain converges quickly.
- The proposal distribution with $\sigma = 0.01$ has the highest acceptance rate but the samples are clustered and the underlying markov chain converges slowly. In the above simulation, even after 10^5 samples (most of which are accepted), the samples are still clustered.

Both of these are undesirable. We need to choose a proposal distribution that has a good balance between the acceptance rate and the rate of convergence.

We see that for $\sigma = 0.5$, the samples are well spread out and the underlying markov chain converges quickly to the target distribution. This is a good choice for the proposal distribution. However, the acceptance rate is 0.07 which is lower than the acceptance rate for naive rejection sampling AND the generated samples are correlated. It is better to use rejection sampling than to use this proposal distribution.

For $\sigma = 0.2$, the acceptance rate is 0.3 which is better than the acceptance rate for naive rejection sampling. The samples are well spread out and the underlying markov chain converges quickly, although not as quickly as for $\sigma = 0.5$. For this example, $\sigma = 0.2$ is a good choice for the proposal distribution. We can fine tune this parameter to get a better acceptance rate, if needed.

Metropolis–Hastings Algorithm

The general algorithm for the Metropolis–Hastings algorithm is an algorithm for generating a sequence of samples from a probability distribution $p(x)$.

Structure of Metropolis–Hastings Algorithm

The above example illustrates the general structure of the Metropolis–Hastings algorithm for sampling from a probability distribution $p(x)$. The algorithm is as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \dots, N-1$:
    1. Generate a proposal y close to x_i.
    2. Evaluate the acceptance ratio $\alpha(y|x_i)$.
    3. Accept y with probability $\alpha(y|x_i)$.
        1. If accepted, set $x_{i+1} = y$.
        2. Else, set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \dots, x_{N-1}$.

Proposal Distribution

For the MH algorithm, we need to specify a **proposal distribution** for generating "nearby points". This is a joint distribution $q(x, y)$ for two random variables X and Y. However, for running the algorithm we only need the conditional distribution $q(y|x)$. As such, it's more common to say that the proposal distribution is the conditional distribution $q(y|x)$. We think of x as the current point and y as the proposed point so that the proposal distribution $q(y|x)$ is *the distribution of the proposed point y given the current point x.*

In the Example 0.1, the proposal distribution $q(y|x)$ was a Gaussian distribution centered at x with a chosen variance,

$$q(y|x) = \mathcal{N}(y; x, \sigma^2 I).$$

The proposal distribution can be any distribution that is easy to sample from. The choice of the proposal distribution is crucial for the efficiency of the algorithm.

Acceptance Criterion

Once we generate a proposal y, we need to evaluate an acceptance criterion for w. The acceptance criterion is based on the ratio of the target distribution $p(x)$ and the proposal distribution $q(y|x)$. We calculate the acceptance ratio as

$$\alpha(y|x) = \min\left\{\frac{p(y)}{p(x)} \cdot \frac{q(x|y)}{q(y|x)}, 1\right\}.$$

The complete Metropolis–Hastings algorithm is as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a proposal y from the proposal distribution $q(y|x_i)$.
    2. Compute the acceptance ratio
    $$\alpha(y|x) = \min\left\{\frac{p(y)}{p(x)} \cdot \frac{q(x|y)}{q(y|x)}, 1\right\}.$$
    3. Accept y with probability $\alpha(y|x)$:
        - If y is accepted, set $x_{i+1} = y$.
        - Otherwise, set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \ldots, x_{N-1}$.

We can expand the algorithm further as follows:

```
1. **Initialization**: Start at a point $x_0$.
2. For $i = 0, 1, \dots, N-1$:
    1. Generate a proposal $y \sim q(y|x_i)$.
    2. Compute
       $$\alpha_1 = p(y) \cdot q(x_i|y), \quad \alpha_2 = p(x_i) \cdot q(y|x_i).$$
    3. If $\alpha_1 \geq \alpha_2$,
        1. Set $x_{i+1} = y$.
    4. Else,
        1. Generate a random number $u \sim \text{Uniform}(0, 1)$.
        2. If $u < \alpha_1 / \alpha_2$,
            1. Set $x_{i+1} = y$.
        3. Else,
            1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \dots, x_{N-1}$.
```

Symmetric Proposal Distribution

Often we use a symmetric proposal distribution $q(y|x) = q(x|y)$. In this case, the acceptance ratio simplifies to

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

The Metropolis algorithm is a special case of the Metropolis–Hastings algorithm where the proposal distribution is symmetric. The Metropolis algorithm is widely used in practice. The algorithm is as follows:

```
1. **Initialization**: Start at a point $x_0$.
2. For $i = 0, 1, \dots, N-1$:
    1. Generate a proposal $y \sim q(y|x_i)$.
    2. If $p(y) \geq p(x_i)$,
        1. Set $x_{i+1} = y$.
    3. Else,
        1. Generate a random number $u \sim \text{Uniform}(0, 1)$.
        2. If $u < p(y)/p(x_i)$,
            1. Set $x_{i+1} = y$.
```

    3. Else,
        1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \dots, x_{N-1}$.

Example 0.2. If $p(x)$ is a uniform distribution over the region Ω, then the Metropolis algorithm reduces to the random walk Metropolis–Hastings algorithm. In this case, the acceptance ratio becomes

$$\alpha(y|x) = \begin{cases} 1, & \text{if } y \in \Omega, \\ 0, & \text{if } y \notin \Omega. \end{cases}$$

This means that we always accept a proposal y if it is in the region Ω and reject it otherwise. This is equivalent to the random walk Metropolis–Hastings algorithm in Section .

Several other algorithms, such as rejection sampling, random walks on graphs, and Gibbs sampling, can be viewed as special cases of the Metropolis–Hastings algorithm.

Metropolis Markov Chain

If the target distribution $p(x)$ is defined over the sample space Ω, then the Metropolis–Hastings algorithm generates a Markov chain with state space Ω.

For simplicity,

1. we will analyze the Metropolis algorithm with a symmetric proposal distribution $q(y|x) = q(x|y)$, and
2. we will assume that the target distribution $p(x)$ is defined over a finite sample space Ω.

The transition matrix for the Markov chain is given by

$$\mathbb{P}(X = b|X = a) = \begin{cases} q(b|a) & \text{if } p(b) \geq p(a) \text{ and } a \neq b, \\ \frac{p(b)}{p(a)} \cdot q(b|a) & \text{if } p(b) < p(a) \text{ and } a \neq b, \\ q(a|a) + \sum\limits_{b \in \Omega, p(b) < p(a)} q(b|a) \cdot \left(1 - \frac{p(b)}{p(a)}\right) & \text{if } a = b. \end{cases}$$

One can check that this defines a valid transition matrix by showing that the sum of the transition probabilities for each state is 1.

$$\begin{aligned}
&\sum_{b\in\Omega} \mathbb{P}(X=b|X=a) \\
&= \sum_{p(b)\geq p(a), a\neq b} q(b|a) + \sum_{p(b)<p(a)} \frac{p(b)}{p(a)} \cdot q(b|a) + q(a|a) + \sum_{p(b)<p(a)} q(b|a)\cdot\left(1-\frac{p(b)}{p(a)}\right) \\
&= \sum_{b\in\Omega} q(b|a) \\
&= 1.
\end{aligned}$$

Detailed Balance Equations

Theorem 0.1. *The Metropolis algorithm is reversible with respect to the target distribution $p(x)$. This means that the Markov chain defined by the Metropolis algorithm satisfies the detailed balance equations:*

$$p(a)\cdot\mathbb{P}(X=b|X=a) = p(b)\cdot\mathbb{P}(X=a|X=b).$$

Proof. We only need check the detailed balance equations for the case when $a \neq b$. The case when $a = b$ is trivial. We make two cases:

1. **Case 1**: $p(a) \neq p(b)$.

Without loss of generality, assume that $p(a) < p(b)$. Then we have

$$\begin{aligned}
p(a)\cdot\mathbb{P}(X=b|X=a) &= p(a)\cdot q(b|a) \\
&= p(a)\cdot q(a|b) \\
&= p(b)\cdot q(a|b)\frac{p(a)}{p(b)} \\
&= p(b)\cdot\mathbb{P}(X=a|X=b).
\end{aligned}$$

2. **Case 2**: $p(a) = p(b)$. In this case, we have

$$\begin{aligned}
p(a)\cdot\mathbb{P}(X=b|X=a) &= p(a)\cdot q(b|a) \\
&= p(b)\cdot q(a|b) \\
&= p(b)\cdot\mathbb{P}(X=a|X=b).
\end{aligned}$$

Thus, in both cases, we have shown that the detailed balance equations hold. This completes the proof.

□