How Markov Chain Monte Carlo (MCMC) Works

Markov Chain Monte Carlo (MCMC) is a class of algorithms used to **sample from a probability distribution** for which direct sampling is difficult or impossible. This is often the case for complex, high-dimensional distributions, especially the **posterior distribution** in Bayesian inference.

The core idea behind MCMC is to construct a **Markov chain** – a sequence of random variables where the probability of transitioning to the next state depends only on the current state.[12] This Markov chain is designed such that its **stationary distribution** (the distribution it converges to after a large number of steps) is the target probability distribution we want to sample from.[3]

Here's a breakdown of the key steps involved:

1. **Define the Target Distribution:** This is the probability distribution we want to sample from (e.g., the posterior distribution in Bayesian inference).[4] We usually know a function that is proportional to this distribution, even if we don't know the normalization constant. Let's call this unnormalized target distribution $f(x)$. Our goal is to get samples that look like they came from $P(x)=Zf(x)$, where $Z=\int f(x)dx$ (the normalization constant) might be unknown.
2. **Initialize:** Start at an arbitrary initial state $x0$ in the sample space of the target distribution.[5]
3. **Propose a New State:** From the current state xt, propose a new candidate state $x*$ according to a **proposal distribution** $Q(x*|xt)$.[6] The choice of the proposal distribution is crucial for the efficiency of the MCMC algorithm.[7] Common choices include:
    - **Random Walk:** Proposing a new state by taking a random step from the current state (e.g., adding a random value from a normal distribution).
    - **Independent Proposal:** Proposing a new state independently of the current state (though this requires some knowledge of the target distribution to be efficient).
4. **Accept or Reject the Proposed State:** Decide whether to move to the proposed state $x*$ or stay at the current state xt.[8] This decision is made based on an **acceptance probability** $\alpha(x*|xt)$.[9] The acceptance probability is designed to ensure that the Markov chain converges to the target distribution. A common acceptance criterion is the **Metropolis-Hastings algorithm**:
   $\alpha(x*|xt)=\min(1,f(xt)Q(x*|xt)f(x*)Q(xt|x*))$
    - The ratio $f(xt)f(x*)$ compares the "likelihood" of the proposed state to the current state under the target distribution.
    - The ratio $Q(x*|xt)Q(xt|x*)$ accounts for the asymmetry in the proposal distribution (how likely it is to propose $x*$ from xt versus proposing xt from $x*$). If the proposal distribution is symmetric (i.e., $Q(x*|xt)=Q(xt|x*)$), the acceptance probability simplifies to:
      $\alpha(x*|xt)=\min(1,f(xt)f(x*))$
5. **Update the State:**
    - Generate a random number u from a uniform distribution between 0 and 1 ($U(0,1)$).[10]
    - If $u\le\alpha(x*|xt)$, then **accept** the proposed state and set $xt+1=x*$.
    - If $u>\alpha(x*|xt)$, then **reject** the proposed state and stay at the current state, setting $xt+1=xt$.
6. **Repeat:** Continue steps 3-5 for a large number of iterations.[11]
7. **Collect Samples:** After a "burn-in" period (initial iterations discarded to allow the chain to converge to the stationary distribution), the sequence of accepted states $(xburn-in+1,xburn-in+2,...,xT)$ forms a sample from the target distribution.

Why does this work?

The acceptance probability is carefully constructed to satisfy the **detailed balance condition** for the Markov chain with respect to the target distribution. Detailed balance ensures that the rate of transitioning from state x to state y is equal to the rate of transitioning from state y to state x under the target distribution. If detailed balance holds, then the target distribution is the stationary distribution of the Markov chain.[12]

Example: Sampling from a Beta Distribution using Metropolis-Hastings

Let's say we want to sample from a Beta distribution with parameters $\alpha=2$ and $\beta=5$. The probability density function (PDF) of a Beta distribution is:

$P(x|\alpha,\beta)=B(\alpha,\beta)x\alpha-1(1-x)\beta-1,\text{for } 0\leq x\leq 1$

where $B(\alpha,\beta)$ is the Beta function (the normalization constant). Let $f(x)=x\alpha-1(1-x)\beta-1$ be the unnormalized target distribution.

We will use a simple **random walk proposal** where we propose a new state $x*$ by taking a step from the current state xt:

$x*=xt+\epsilon$

where ϵ is drawn from a symmetric distribution centered at zero, such as a uniform distribution $[-0.1,0.1]$. Since the proposal distribution is symmetric ($Q(x*|xt)=Q(xt|x*)$), the acceptance probability simplifies to:

$\alpha(x*|xt)=\min(1,f(xt)f(x*))=\min(1,(xt)2-1(1-xt)5-1(x*)2-1(1-x*)5-1)=\min(1,xt(1-xt)4x*(1-x*)4)$

Steps:

1. **Initialize:** Let's start with $x0=0.5$.
2. **Propose:** Draw ϵ from $U(-0.1,0.1)$. Let's say $\epsilon=0.05$. Then $x*=0.5+0.05=0.55$.
3. Calculate Acceptance Probability:
   $f(xt)=0.5(1-0.5)4=0.5\times0.0625=0.03125$
   $f(x*)=0.55(1-0.55)4=0.55\times(0.45)4=0.55\times0.04100625\approx0.02255$
   $\alpha(0.55|0.5)=\min(1,0.031250.02255)=\min(1,0.7216)=0.7216$
4. **Accept or Reject:** Generate a random number u from $U(0,1)$. Let's say $u=0.3$. Since $u=0.3\leq0.7216$, we **accept** the proposed state. So, $x1=0.55$.
5. **Repeat:** We continue this process for many iterations. The sequence of accepted x values will eventually approximate samples from the Beta(2, 5) distribution.

Important Considerations:

- **Burn-in Period:** The initial samples of the Markov chain might be far from the stationary distribution, so they are typically discarded.[13]
- **Mixing:** The rate at which the Markov chain converges to the stationary distribution is crucial. A chain that mixes slowly will require many more iterations to produce representative samples.
- **Proposal Distribution:** The choice of the proposal distribution significantly affects the efficiency of the MCMC algorithm.[14] A good proposal distribution will lead to a reasonable acceptance rate and good mixing.[15]
- **Autocorrelation:** Successive samples in an MCMC chain are often correlated.[16] Techniques like thinning (keeping only every n-th sample) can be used to reduce autocorrelation.[17]

MCMC algorithms are powerful tools for sampling from complex distributions and are widely used in Bayesian statistics, computational physics, and other fields where direct sampling is intractable.[18] Common MCMC algorithms include Metropolis-Hastings, Gibbs sampling, and Hamiltonian Monte Carlo.