STA 3180 Statistical Modelling: Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC)

Definition

Markov Chain Monte Carlo (MCMC) is a class of algorithms used to sample from a probability distribution. It works by constructing a Markov Chain whose stationary distribution is the desired distribution. MCMC algorithms are used for Bayesian inference and can be used to approximate expectations with respect to complicated distributions.

Key Concepts

- Markov Chains: A Markov Chain is a stochastic process that satisfies the Markov property. This means that the future state of the process depends only on its current state, not on its past states.

- Stationary Distribution: A stationary distribution is a probability distribution that does not change over time.

- Sampling: Sampling is the process of randomly selecting observations from a population.

- Bayesian Inference: Bayesian inference is a method of statistical inference in which the prior distribution is updated using data to obtain the posterior distribution.

Coding Examples

Example 1: Metropolis-Hastings Algorithm

```
Start of Code

import numpy as np

def metropolis_hastings(x0, f, proposal_dist, n_iter):
    """
    Implements the Metropolis-Hastings algorithm.

    Parameters
    ----------
    x0 : float
        Initial value of the Markov Chain.
    f : function
        Target distribution.
    proposal_dist : function
        Proposal distribution.
    n_iter : int
        Number of iterations.

    Returns
    -------
```

```
    samples : array
        Array of samples from the target distribution.
    """
    # Initialize the Markov Chain
    x = x0
    samples = [x]
    # Iterate n_iter times
    for i in range(n_iter):
        # Sample from the proposal distribution
        x_prop = proposal_dist(x)
        # Compute the acceptance probability
        alpha = min(1, (f(x_prop)/f(x)))
        # Sample a uniform random variable
        u = np.random.uniform()
        # Update the Markov Chain
        if u < alpha:
            x = x_prop
        samples.append(x)
    return np.array(samples)
End of Code
```

Example 2: Gibbs Sampling

```
Start of Code
import numpy as np
def gibbs_sampling(x0, f, proposal_dist, n_iter):
    """
    Implements the Gibbs Sampling algorithm.
    Parameters
    ----------
    x0 : array
        Initial values of the Markov Chain.
    f : function
        Target distribution.
    proposal_dist : list of functions
        List of proposal distributions.
    n_iter : int
        Number of iterations.
    Returns
    -------
    samples : array
        Array of samples from the target distribution.
    """
    # Initialize the Markov Chain
    x = x0
    samples = [x]
    # Iterate n_iter times
```

```
    for i in range(n_iter):
        # Sample from the proposal distributions
        for j in range(len(x)):
            x[j] = proposal_dist[j](x)

        # Compute the acceptance probability
        alpha = min(1, (f(x)/f(x0)))

        # Sample a uniform random variable
        u = np.random.uniform()

        # Update the Markov Chain
        if u < alpha:
            x0 = x
        samples.append(x)

    return np.array(samples)
```

End of Code