STA 3180 Statistical Modelling: Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC)

Definition

Markov Chain Monte Carlo (MCMC) is a type of computational algorithm used to generate samples from a probability distribution. It is based on the concept of a Markov chain, which is a sequence of random variables that have a certain probability of transitioning from one state to another. MCMC algorithms are used in Bayesian inference and statistical modelling to approximate complex distributions and solve difficult problems.

Key Concepts

- **Markov Chain**: A Markov chain is a sequence of random variables that have a certain probability of transitioning from one state to another. The transition probabilities depend only on the current state and not on the previous states.

- **Markov Chain Monte Carlo (MCMC)**: MCMC is a type of computational algorithm used to generate samples from a probability distribution. It is based on the concept of a Markov chain.

- **Metropolis-Hastings Algorithm**: The Metropolis-Hastings algorithm is a popular MCMC algorithm. It is used to generate samples from a target distribution by constructing a Markov chain whose stationary distribution is the target distribution.

- **Gibbs Sampling**: Gibbs sampling is a special case of the Metropolis-Hastings algorithm. It is used to generate samples from a target distribution by constructing a Markov chain whose stationary distribution is the target distribution.

Coding Examples

Example 1: Metropolis-Hastings Algorithm

```
Start of Code
import numpy as np
def metropolis_hastings(target_dist, initial_state, num_samples):
        # Initialize the Markov chain
        chain = [initial_state]
        # Generate samples
        for i in range(num_samples):
                # Propose a new state
                proposed_state = propose_state(chain[-1])
```

```
            # Calculate the acceptance probability
            acceptance_prob = min(1, target_dist(proposed_state) /
            target_dist(chain[-1]))
            # Sample a uniform random variable
            u = np.random.uniform()
            # Accept or reject the proposed state
            if u < acceptance_prob:
                  chain.append(proposed_state)
            else:
                  chain.append(chain[-1])
      return chain
End of Code
```

Example 2: Gibbs Sampling

```
Start of Code
import numpy as np
def gibbs_sampling(target_dist, initial_state, num_samples):
      # Initialize the Markov chain
      chain = [initial_state]
      # Generate samples
      for i in range(num_samples):
            # Sample a uniform random variable
            u = np.random.uniform()
            # Propose a new state
            proposed_state = propose_state(chain[-1], u)
            # Accept the proposed state
            chain.append(proposed_state)
      return chain
End of Code
```