# STA 3180 Statistical Modelling: Markov Chain Monte Carlo

# STA 3180 Statistical Modelling - Lecture Notes on Markov Chain Monte Carlo

## Introduction

Markov Chain Monte Carlo (MCMC) is a powerful tool for statistical modelling. It is a type of computational algorithm used to sample from a probability distribution. MCMC algorithms are used to simulate complex systems and to estimate parameters in statistical models. MCMC algorithms are based on the Markov Chain, which is a stochastic process that has the property of memorylessness. This means that the future state of the system depends only on its current state, not on the states that preceded it.

## Key Concepts

The key concepts of MCMC include:

- Markov Chains: A stochastic process with the property of memorylessness.

- Monte Carlo Simulation: A technique for estimating the value of a function by randomly sampling from its domain.

- Metropolis-Hastings Algorithm: An MCMC algorithm for sampling from a probability distribution.

- Gibbs Sampling: An MCMC algorithm for sampling from a multivariate probability distribution.

## Definitions

- **Markov Chain**: A stochastic process in which the future state of the system depends only on its current state, not on the states that preceded it.

- **Monte Carlo Simulation**: A technique for estimating the value of a function by randomly sampling from its domain.

- **Metropolis-Hastings Algorithm**: An MCMC algorithm for sampling from a probability distribution. It works by proposing a new state for the system and accepting or rejecting it based on a probability.

- **Gibbs Sampling**: An MCMC algorithm for sampling from a multivariate probability distribution. It works by iteratively sampling from the conditional distributions of each variable given the values of the other variables.

## Coding Examples

### Example 1: Metropolis-Hastings Algorithm

Start of Code

```python
def metropolis_hastings(x, pdf, proposal_pdf, num_samples):
        samples = []
        for _ in range(num_samples):
                x_new = sample_from_proposal(x, proposal_pdf)
                acceptance_prob = min(1, pdf(x_new) / pdf(x))
                if np.random.uniform() < acceptance_prob:
                        x = x_new
                samples.append(x)
        return samples
```

End of Code

This code implements the Metropolis-Hastings algorithm. The function takes four arguments: `x`, which is the current state of the system; `pdf`, which is the probability density function of the target distribution; `proposal_pdf`, which is the probability density function of the proposal distribution; and `num_samples`, which is the number of samples to generate. The function returns a list of samples from the target distribution.

### Example 2: Gibbs Sampling

Start of Code

```python
def gibbs_sampling(x, pdf, num_samples):
        samples = []
        for _ in range(num_samples):
                for i in range(len(x)):
                        x[i] = sample_from_conditional(x, i, pdf)
                samples.append(x)
        return samples
```

```
End of Code
```

This code implements the Gibbs Sampling algorithm. The function takes three arguments: `x`, which is the current state of the system; `pdf`, which is the joint probability density function of the target distribution; and `num_samples`, which is the number of samples to generate. The function returns a list of samples from the target distribution.