# Sampling

## Part I: Proposal Distribution Calculation

For the Gibbs sampling method, we need to define the proposal distribution that determines how the Markov chain will transition between states. Since we are dealing with a Markov random field where each variable node is only connected to its neighbors, the proposal distribution for updating a particular variable $X_i$ can be derived from the joint distribution *p*($X$) as shown in equation 8.49 in the PRML textbook:

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N). \qquad (8.49)$$

To compute the conditional probability *p(*$X_i$ | $X_1$, ..., $X_{i-1}$, $X_{i+1}$, ..., $X_N$*)* needed for the Gibbs sampler, we can use the property that $X_i$ only depends on its immediate neighbors $X_{i-1}$ and $X_{i+1}$ due to the Markov property. This means we can rewrite the joint distribution as follows:

$$p(X) = \frac{1}{Z}\ \psi_{i-1,i}(X_{i-1}, X_i)\ \psi_{i,i+1}(X_i, X_{i+1})$$

By dividing both sides by the terms that do not involve $X_i$ we get the following conditional probability:

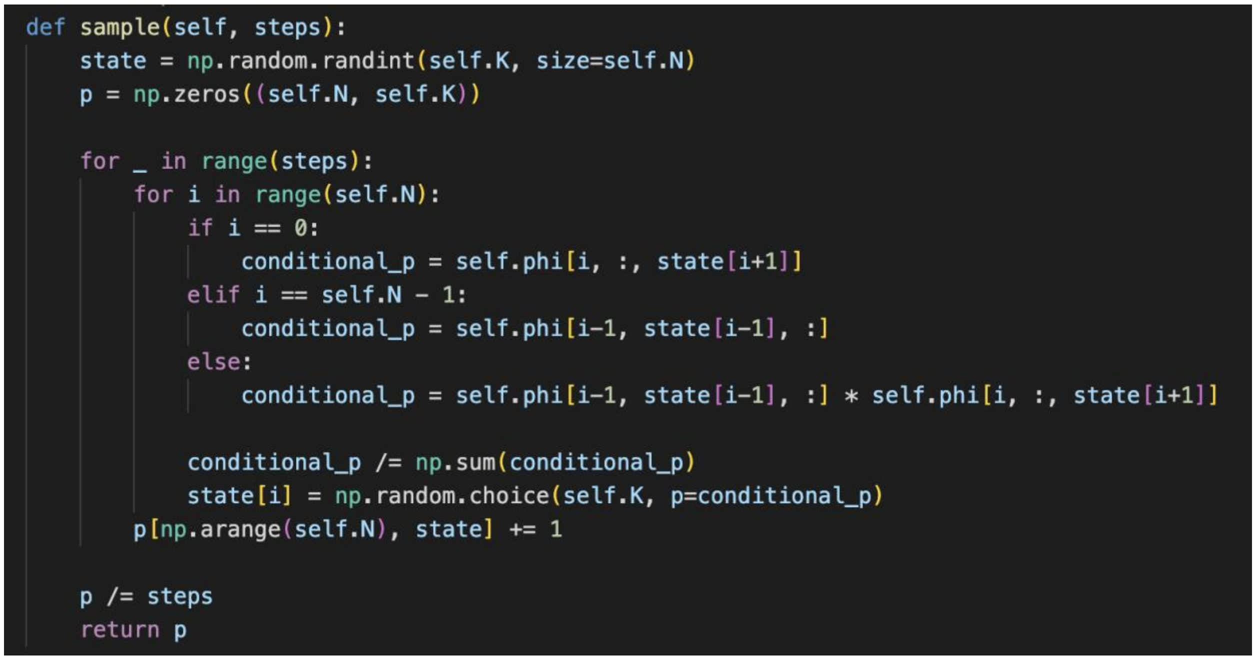
$$p(X_i|X_{i-1}, X_{i+1}) = \frac{1}{Z_i}\ \psi_{i-1,i}(X_{i-1}, X_i)\ \psi_{i,i+1}(X_i, X_{i+1})$$

Where $Z_i$ is the normalization constant that ensures the conditional probability sums to 1 over all possible values of $X_i$.

*Code Implementation*

In this code snippet below, *conditional_p* represents the conditional probability distribution $p(X_i|X_{i-1}, X_{i+1})$for variable $X_i$. Depending on the position of $X_i$in the chain, the conditional probability is computed differently:

- If $X_i$ is the first variable, the proposal distribution is based only on the potential function associated with $X_i$and $X_{i+1}$.
- If $X_i$is the last variable, the proposal distribution is based only on the potential function associated with $X_{i-1}$ and $X_i$.
- Otherwise, the proposal distribution is based on potential functions associated with both $X_{i-1}$ and $X_i$, and $X_i$ and $X_{i+1}$.



```python
def sample(self, steps):
    state = np.random.randint(self.K, size=self.N)
    p = np.zeros((self.N, self.K))

    for _ in range(steps):
        for i in range(self.N):
            if i == 0:
                conditional_p = self.phi[i, :, state[i+1]]
            elif i == self.N - 1:
                conditional_p = self.phi[i-1, state[i-1], :]
            else:
                conditional_p = self.phi[i-1, state[i-1], :] * self.phi[i, :, state[i+1]]

            conditional_p /= np.sum(conditional_p)
            state[i] = np.random.choice(self.K, p=conditional_p)
        p[np.arange(self.N), state] += 1

    p /= steps
    return p
```

*Figure 1: Gibbs Sampling Implementation*

## Part II: Gibbs vs. Exact Accuracy Analysis

To compare the accuracy of the Gibbs sampler approximations with the exact calculations, I used the Mean Squared Error between the results from *sample()* and *exact()* as a comparison metric for a synthetic chain where *N =5* and *K =3*. MSE was calculated as:
*np.mean((exact_prob* - *sampled_prob) ** 2)*

*Quantifying the Relation between Metric and Number of Steps:*
I experimented with varying numbers of steps for the Gibbs sampler, ranging from 1 to 1000 steps. For each step count, MSE was computed and plotted as the number of steps increased. This allows for analysis of the convergence behavior of the Gibbs sampler and determine how quickly it approximates the exact calculations.
Identifying Maximum Steps for Efficient Computation
Since this is a simple chain, *exact()* is relatively fast. Hence, in this case, the largest number of steps that the Gibbs sampler can compute without exceeding the time taken to obtain exact results was 2 steps.
*Supporting Graph*
The graph illustrates the relationship between the number of steps taken by the Gibbs sampler and the value of MSE. This graph visually depicts the convergence behavior of the Gibbs sampler and provides insights into its performance relative to the exact calculations.

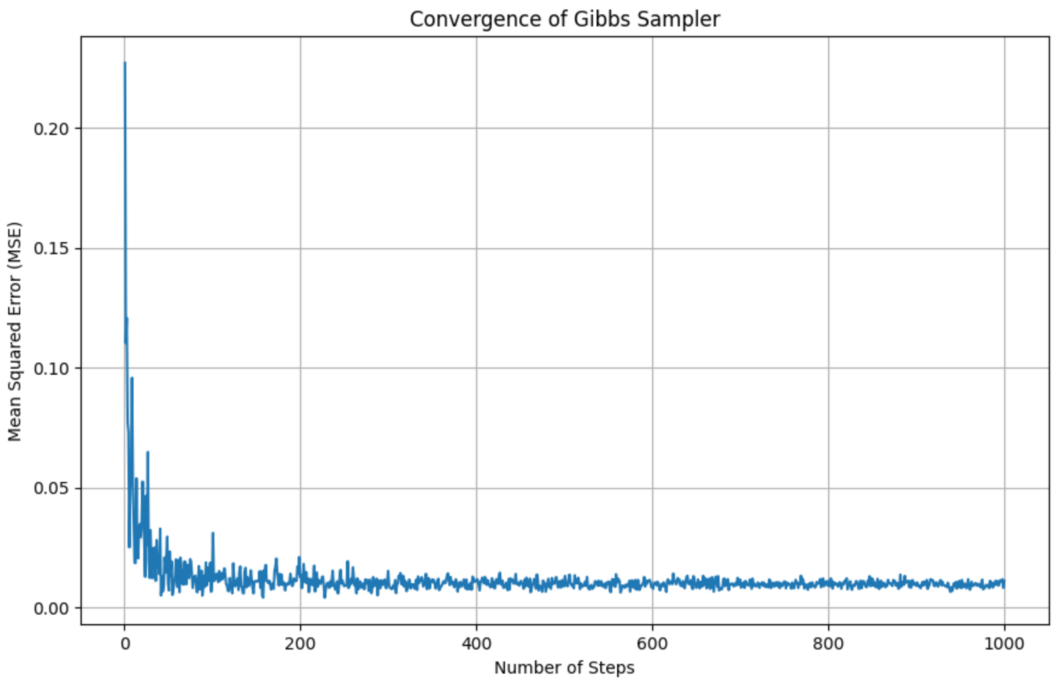


*Figure 2: MSE over various number of steps*

*Conclusion*

Through this analysis, we gain a good understanding of the accuracy and efficiency of the Gibbs sampler in approximating marginal probabilities in the Markov random field. By quantifying the relation between the evaluation metric (MSE) and the number of steps, we can clearly see the convergence behavior of the Gibbs sampler over time and also make informed decisions about the optimal settings for Gibbs sampling in practical applications.