Image Generation using Gibbs Sampling

Advanced ML (CS60075): Programming Assignment 2 Report

Shikhar Mohan (18EC10054)

7 November 2021

1 Abstract

Gibbs sampling belongs to the Markov Chain Monte Carlo (MCMC) class of sampling methods which is particularly suited for drawing samples from distributions which are intractable due to their high dimensionality. In this assignment, we sample from a high dimensional gaussian distribution using the Gibbs sampling method and observe the samples across sampling iterations.

2 Theoretical Background

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. As is characteristic of every MCMC algorithm, in Gibbs sampling we model the probability distribution using a Markov Chain. If we want to obtain k samples of $\mathcal{X} = \{x_1, x_2, ..., x_N\}$, from a joint distribution $p(x_1, x_2, ..., x_N)$, we denote the ith sample by $\mathcal{X}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, ..., x_N^{(i)}\}$ and proceed as follows:

1. We start with an initial sample $\mathcal{X}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, ..., x_N^{(i)}\}$.
2. We want $\mathcal{X}^{(i+1)}$ now. We sample each $x_i^{(i+1)}$ by conditioning on all the information available so far, i.e. more formally, we have:
$$x_i^{(i+1)} \sim p(x_i | x_1^{(i+1)}, .., x_{i-1}^{(i+1)}, x_{i+1}^{(i)}, .., x_N^{(i)})$$
3. Repeat the above k times.

This algorithm works only if the probability function is non-zero everywhere, because only then does the sampling converge due to the ergodicity of our Markov Chain. As done in all MCMC methods, we take into consideration a burn-in phase as well, where we discard the initial transitory samples and consider only the later ones. We take small steps if the variables are correlated (which is not the case in our task). It is also important to note since we obtain each single variable by conditioning on all information available so far, the process is sequential and has no scope of parallelisability. Moreover, high dimensional distributions take much longer to converge both in terms of number of steps as well as the time required to finish each step (due to lack of parallelisability).

3 Observation

In this section we detail our findings when we use Gibbs sampling to draw images from the specified distribution, and we run a few additional experiments as well, with some different images and hyperparameters to gain a fuller understanding.

3.1 Given Task

For our given task, we have μ as our mean vector which is derived from the image M, where $M \in \mathbb{R}^{30\times 30}$, $m_{ij} = (i+j)/100$ and $\Sigma = 0.1I$. Due to the diagonal nature of our covariance matrix, the equation for conditioning simplifies greatly. We simply have:

$$x_i^{(j)} \sim \mathcal{N}(\mu_i, 0.1)$$

Where μ_i is the ith element in the mean vector. We run the sampling algorithm for 40 iterations and observe the sample at every 10th iteration.

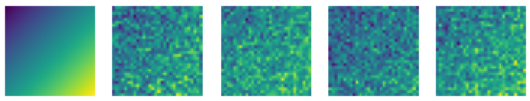

Figure 1: First image is the mean image, rest four are samples generated at 10th, 20th, 30th and 40th iterations.

We observe a slight gradient from the top left to bottom right, which means some structure from the original mean image is preserved but we can see a lot of 'noise' in the samples as well. Moreover, there is no observable convergence as iterations increase.

3.2 Additional Experimentation

We test the Gibbs sampling algorithm on four more images, where we observe the samples at 10th iteration.

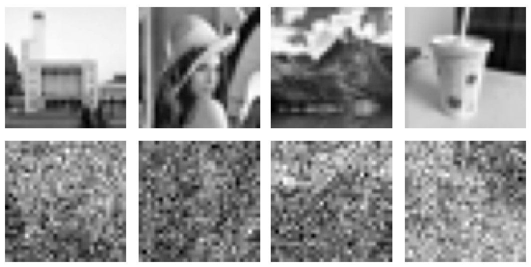

Figure 2: First row belongs to new images, second row belongs to corresponding samples at the 10th iteration.

We can see that there is a lot of noise in the images, but some semblance of the original image structure is preserved. Now we vary the scaling coefficient of the covariance matrix to see how that affects the sampling. Formally, we set $\Sigma = \lambda I$ where $\lambda = \{0.1, 0.01, 0.0025, 10^{-4}\}$ for two images.

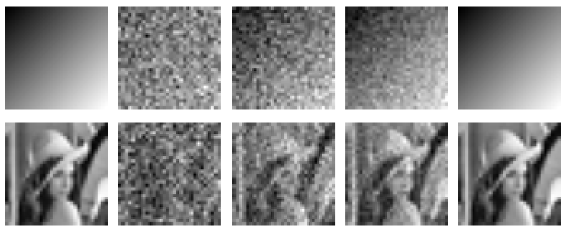

Figure 3: Leftmost images in each row are original, subsequent images correspond to $\lambda = \{0.1, 0.01, 0.0025, 10^{-4}\}$

.

We observe that lower values of the covariance matrix correspond to clearer images, indicating that the algorithm is indeed functional and relies heavily on the covariance matrix for its success.

4 Discussion

We deduce multiple ideas from the above experiments. Namely:

1. Due to the diagonal nature of our covariance matrix, the sampling equation simplifies to $x_i^{(j)} \sim \mathcal{N}(\mu_i, 0.1)$. Since we have no term corresponding to the previous sample in this equation, it means that every iteration generates the same sample, or in other words the best possible sample is generated after the first iteration itself. We can see this clearly in figure 1, where no subsequent iterations posit any sign of convergence.

2. A diagonal covariance matrix with all diagonal elements equal to 0.1 results in every pixel being independently drawn from a gaussian where $\sigma^2 = 0.1$. This means that $\sigma \approx 0.31$, which is the standard deviation. In an image where pixel values lie in the range $[0, 1]$, a standard deviation of nearly 0.31 produces a significant amount of noise since it is easy for the samples to stray off the mean, which we can see in figures 1 and 2. This claim is reinforced further by figure 3 where making diagonal elements in the covariance matrix small makes the sampled image much clearer and closer to the mean.

5 Conclusion

We successfully sample multiple images from the defined high dimensional Gaussian distribution. We observe that the samples do not converge due to the sampling equation being invariant to the iteration number. This property arises from the off-diagonal elements being zero in the covariance matrix. Moreover, we also see that the images contain a lot of noise compared to the mean image due to relatively large values present in the covariance matrix, which we demonstrate to be true by obtaining clearer images closer to the mean by using a covariance matrix with much smaller values.