Gibbs Sampling

Research and Development Project

R. Chidaksh
Mentored By **Prof. Prabuchandran K.J**

November 10, 2022

Index

1 Introduction

The aim of the project is to understand the importance of Gibbs Sampling and exploring the applications of Gibbs Sampling. In this project we started by applying Gibbs Sampling on Guassian distribution and later we applied Gibbs Sampling on Lantent Dirchilet Allocation(LDA).

1.1 Project Components

We can divide the above Project into two components. In the first component we applied Gibbs Sampling on Guassian Distributions. In the second component we applied Gibbs Sampling on Dirchilet Distributions.

2 Gibbs Sampling

Gibbs Sampling is an MCMC sampling method where we construct a Markov chain which is used to sample from a desired joint (conditional) distribution. Let's say we have a probability distribution p(X) where X is a K dimensional Random Variable. Say if p(x) is too complex or K is very large then it becomes very difficult to sample from the original distribution p(x) directly. So we use Gibbs Sampling where we sample each x_k from the conditional distribution of x_k given $x_{\sim k}$ where $x_{\sim k}$ denotes all other variables except k and we repeat this iteratively until we are confident enough that the final samples are from p(x). The pseudo code for the above process is described below.

Algorithm 1 Basic Gibbs Sampling Process

```
procedure GIBBS SAMPLER
    Randomly Initialize x_0, x_1, x_2...x_k
    for k = 1, 2, .... do
        for i = 1, 2, ....n do
            Draw x_i ~ P(x_i | x_~i)
        end for
        X = (x_1, ....x_k)
    end for
end procedure
```

3 First Component

3.1 Gibbs Sampling in finding solution to Linear System of equations

There are many methods to solve a linear system of equations like the Matrix Inversion method [5] and Cramer's method [2], involves calculating the product of two matrices. Matrix Multiplication is computationally intensive since the complexity of multiplying two matrices, say $A(m \times n)$ with $B(n \times p)$ is of Order $O(m \times n \times p)$. So for two square matrices of Order $n \times n$ is $O(n^3)$. This becomes very inefficient in large dimension matrices when using methods like matrix inversion to find the solution to the linear system of equations.

Methods like the steepest descent [6], conjugate gradient [6] and Jacobi Method [1] solve this issue by multiplying vectors instead of matrices which reduces the complexity to $O(n^2)$. In this work, we explore Gibbs Sampling as an alternative method to solve a linear system of equations evading computationally intensive matrix multiplications.

3.2 Method

Before solving the actual problem we tried using Gibbs sampling to sample from a Multi-Variate Normal(MVN) given mean and inverse of covariance matrix without finding the covariance matrix. Say we are dealing with n-dimensional multivariate which means Mean(μ) is a $n \times 1$ matrix and covariance (Σ) is a symmetric positive definite matrix of order $n \times n$ so Σ^{-1} say A is also symmetric and positive definite of order $n \times n$. Our matrix X is a vector $[x_1, x_2, ..., x_n]$, we can break X into two parts X_1, X_2 where X_2 contains only one element whereas X_1 contains $X - X_2$ i.e all elements in X except X_2. Let $X_1 \sim N(\mu_1, \Sigma_{11})$ and $X_2 \sim N(\mu_2, \Sigma_{22})$ then we can split Σ into block matrices,

$$\Sigma = \left[\begin{array}{c|c} \Sigma_{11} & \Sigma_{12} \\ \hline \Sigma_{21} & \Sigma_{22} \end{array}\right].$$

and let $x = [x1, x2..x_{i-1}, x_{i+1}, ...x_n]$ then,

$$E[X_2|X_1 = x] = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x - \mu_1)$$
$$\Sigma_{22|1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$

But we don't have Σ we have only A so to get the block matrices of Σ from A we use Schur complement of matrix Σ. Schur Complement [7] of a sample block matrix P, below $S = D - CA^{-1}B$

$$P = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{bmatrix}.$$

Here in place of P, we have our Σ. We can use the matrix $A = \Sigma^{-1}$ to get the terms $\Sigma_{21}\Sigma_{11}^{-1}$ and $\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$ directly without doing any matrix multiplications. After getting those terms we can update X_2,

$$X_2 \sim N(E[X_2|X_1 = x], \Sigma_{22|1})$$

and thus we will iteratively modify each dimension of X and generate our samples.

3.3 Algorithm

Start at a random n dimensional point X_0 vector with attributes $x_1, x_2, x_3, ..., x_n$ and we have inverse of the covariance matrix A. For simplicity we are taking these X_i are sampled from $N(0, \Sigma)$.

Algorithm 2 Gibbs Sampler

procedure GIBBS SAMPLER
    $X_0 \leftarrow (x_0, x_1, x_2...x_n)$
    for i *in* $range(1, n)$ **do**
        for i *in* $range(n)$ **do**
            $temp \leftarrow (x_0, x_1, ..x_{i-1}, x_{i+1}, ...x_n)$

            $mean \leftarrow \dfrac{A[i][:].delete(A[i][i])}{-A[i][i]}.temp$

            $var \leftarrow 1/A[i][i]$
            $x_i \leftarrow N(mean, var)$
        end for
        $X_i \leftarrow (x_0, x_1, x_2, ...x_n)$
    end for
end procedure

Algorithm 3 Check Gibbs Sampler

procedure CHECK GIBBS SAMPLER
  $sum \leftarrow$ Null matrix
  for i *in* $range(n)$ **do**
    $sum \leftarrow \frac{sum * i + x_i.{x_i}^T}{i+1}$
  end for ▷ $inv(A)$ is inverse of A
  $Euclidean\ norm = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(sum[i][j] - inv(A)[i][j])^2}$
end procedure

For large n, the matrix *sum* should be converging to the covariance matrix inv(A) or inverse of A, which implies our euclidean norm should be approaching zero as n tends to infinity.

3.4 Experiments

Many experiments were done starting from comparing the steepest descent and conjugate gradient to find the solution of the linear system of equations to sampling from a multivariate normal distribution with known mean and inverse of the covariance matrix using Gibbs sampling without actually computing the inverse of the covariance matrix

3.4.1 Experiment for comparing steepest descent and conjugate gradient

we compared the two algorithms in the number of iterations and time they take before converging to a solution with an error less than 10^-9 while solving the linear system of equations $Ax = b$ where A, b, x_0(initial starting point) were generated randomly. For a fixed dimension, the value of A, b, x_0 were kept the same for both the steepest descent and Conjugate Gradient algorithm. Here A is of Order $n \times n$,b is of Order $n \times 1$ and x is order $n \times 1$

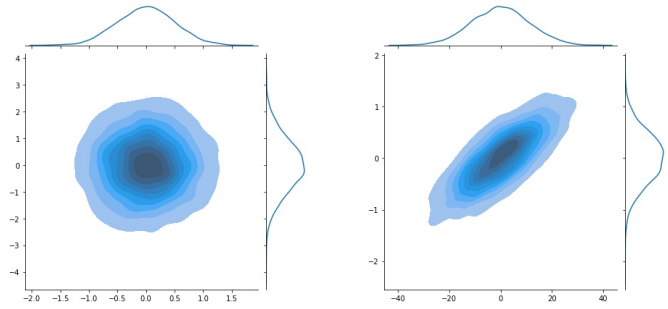
	Steepest Descent		Conjugate Gradient	
n	iterations	time (in sec)	iterations	time (in sec)
10	507682	10.5	10	338μ
50	5419350	117	50	$1.61m$
100(sparse)	2490	$95.3m$	64	$16.5m$

Methods like steepest descent and conjugate gradient work for large dimensions but the conjugate gradient is better because it converges in less number of iterations and time, especially in the case of sparse matrices whereas the positive side of steepest descent is it can converge a bit closer to the actual value than conjugate gradient due to more number of iterations and time taken per iteration

is less in steepest descent. Plain steepest descent takes many iterations but with certain optimizations, we can make the steepest descent converge in less number of iterations. [4]

3.4.2 Experiment to sample from bivariate and draw the KDE contour plots

We sampled from a bivariate with a Mean(μ) and covariance(Σ) using the Cholesky decomposition [3]. For any affine transformation of $X \sim N(0, I)$ to $Y = ZX + M$ then $Y \sim N(M, Z.(Z.T))$ but here we want samples with Mean μ so we keep $M = \mu$ and covariance Σ so $Z.(Z.T) = \Sigma$ which is nothing but Cholesky decomposition [3] of Σ. Hence here $Z = Cholesky(\Sigma)$. In this experiment for simplicity, we took Mean as zero, $\mu = 0$. We get circular contours for bivariate when the two random variables are uncorrelated. When they are correlated, we get elliptical contours.



(a) the plot of bivariate with zero mean and zero covariance

(b) the plot of bivariate with zero mean and non-zero covariance

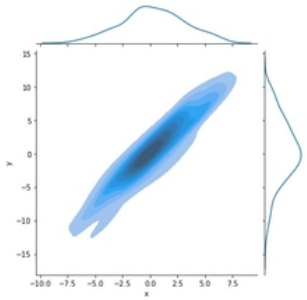
Figure 1: bivariate joint density KDE plots

Contour KDE plots of two gaussian random variables which are uncorrelated are circular. But when we add correlation between those two random variables, the KDE contour plots become elliptical. We can also say that each circular contour C with a radius r has its corresponding elliptical contour E related by $E = LC$, where L is the Cholesky decomposition [3] of Σ

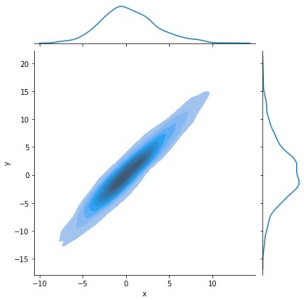
3.4.3 Experiment verifying the Gibbs sampler

The samples drawn from the Gibbs sampler were plotted to check whether they match with the KDE contour plots for a normal bivariate. We took 100,1000,10000 samples and plotted the respective KDE contour plots for all the samples. For the bivariate, we took Mean $\mu = 0$ and the covariance matrix

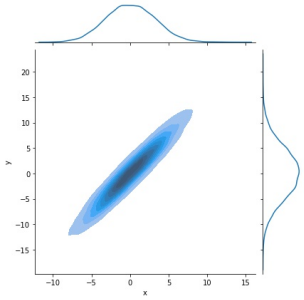
$$\Sigma = \begin{bmatrix} 10.10549468 & 15.08509136 \\ 15.08509136 & 24.89129378 \end{bmatrix}.$$



(a) joint bivariate KDE plot of scatter points(100) sampled using Gibbs sampler



(b) joint bivariate KDE plot of scatter points(1000) sampled using Gibbs sampler



(c) joint bivariate KDE plot of scatter points(10000) sampled using Gibbs sampler

Figure 2: Gibbs sampler plots

The KDE contour plots for all the three plots match with the standard bivariate KDE plots which shows that our sampler is indeed generating samples from bivariate normal with a given distribution but the error in the covariance matrix generated from the samples drawn to the actual covariance matrix varies with the number of samples.

3.4.4 Gibbs Sampler for linear system of equations

For the above task we took three sparse Stiffness matrices of dimension 100×100, 1225×1225 and 10000×10000 and corresponding Force matrices of dimension 100×1, 1225×1 and 10000×1. According to the method specified above we implemented Gibbs Sampling and we used different metrics to check the convergence of the Solution matrix(X) obtained by Gibbs sampling. Our convergence test was to check how close the samples are to A^{-1} where A is the stiffness matrix. In our matrix X_i as we shown earlier in the Check Gibbs Sampler Algorithm, our sum matrix should be close to A^{-1}.

We implemented the following convergence techniques,

- Plane Gibbs Sampler Convergence:

$$sum = \frac{sum * i + x_i.{x_i}^T}{i+1}$$

  Where we used all the samples obtained in the process and check the convergence.

- Conjugate Norm:

$$sum = sum + (x_k.T * x_k)/(x_k.T * A * x_k)$$

  Where we used X after updating all it's k entries. Then weighted the sum according to number of samples.

- Jump Sigma Convergence: It's same as Plane Sigma Convergence but the samples used to calculate sum are taken after updating all the k entries of X.

- Sigma Conjugate Sigma Convergence: Here we took sum as

$$sum = \alpha \times sum_1 + 1 - \alpha \times sum_2$$

  where sum_1 is sum value obtained in Jump Sigma Convergence
  where sum_2 is sum value obtained in Conjugate Sigma Convergence and α is a constant between $(0, 1)$

3.4.5 Updating all entries at a time

We tried to update all entries at a time to speed up the sampling process but it didn't work as it was giving large norm differences with higher dimensions after certain number of iterations.

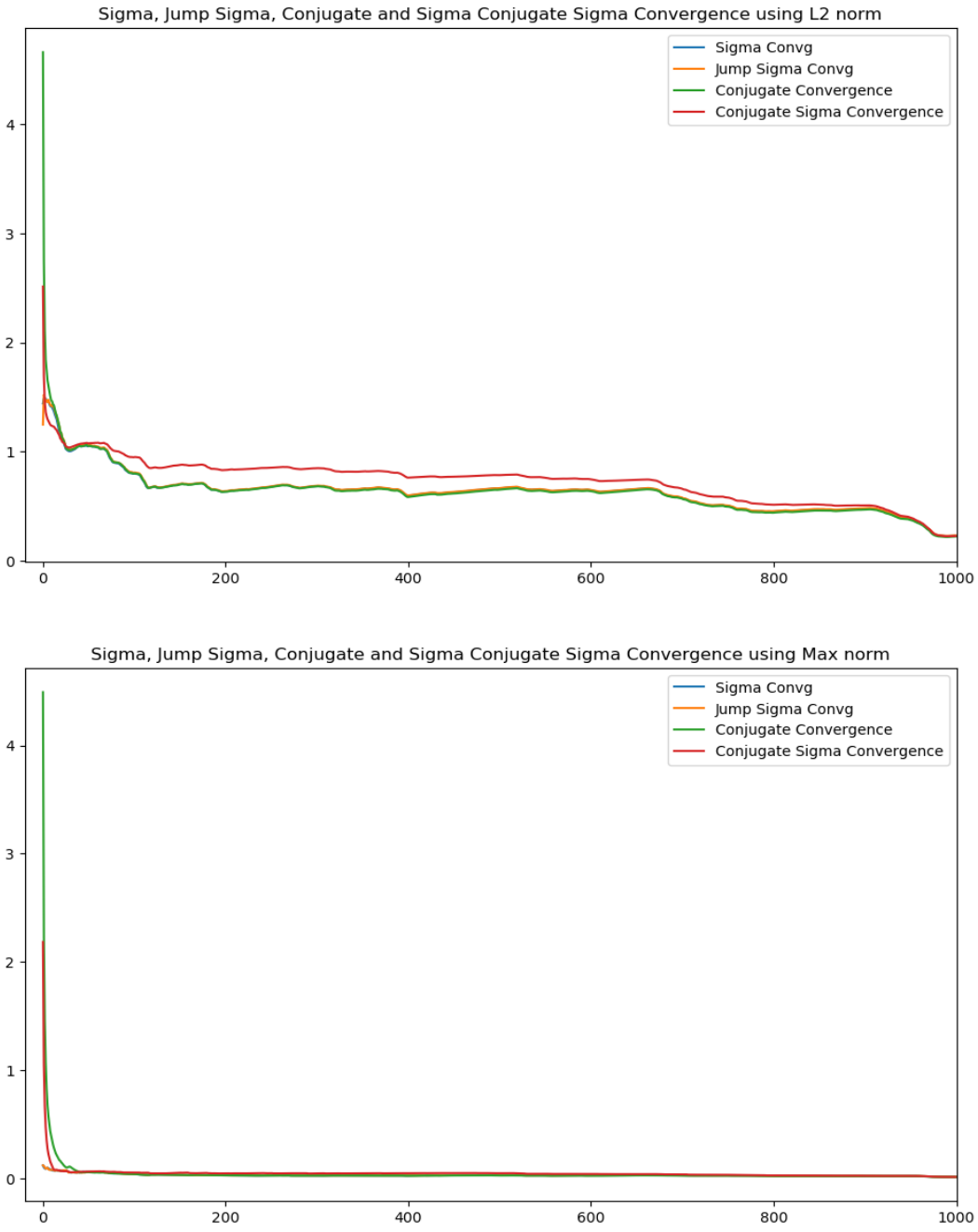


Figure 3: Comparision between Convergence techniques for 100 dimensional matrix

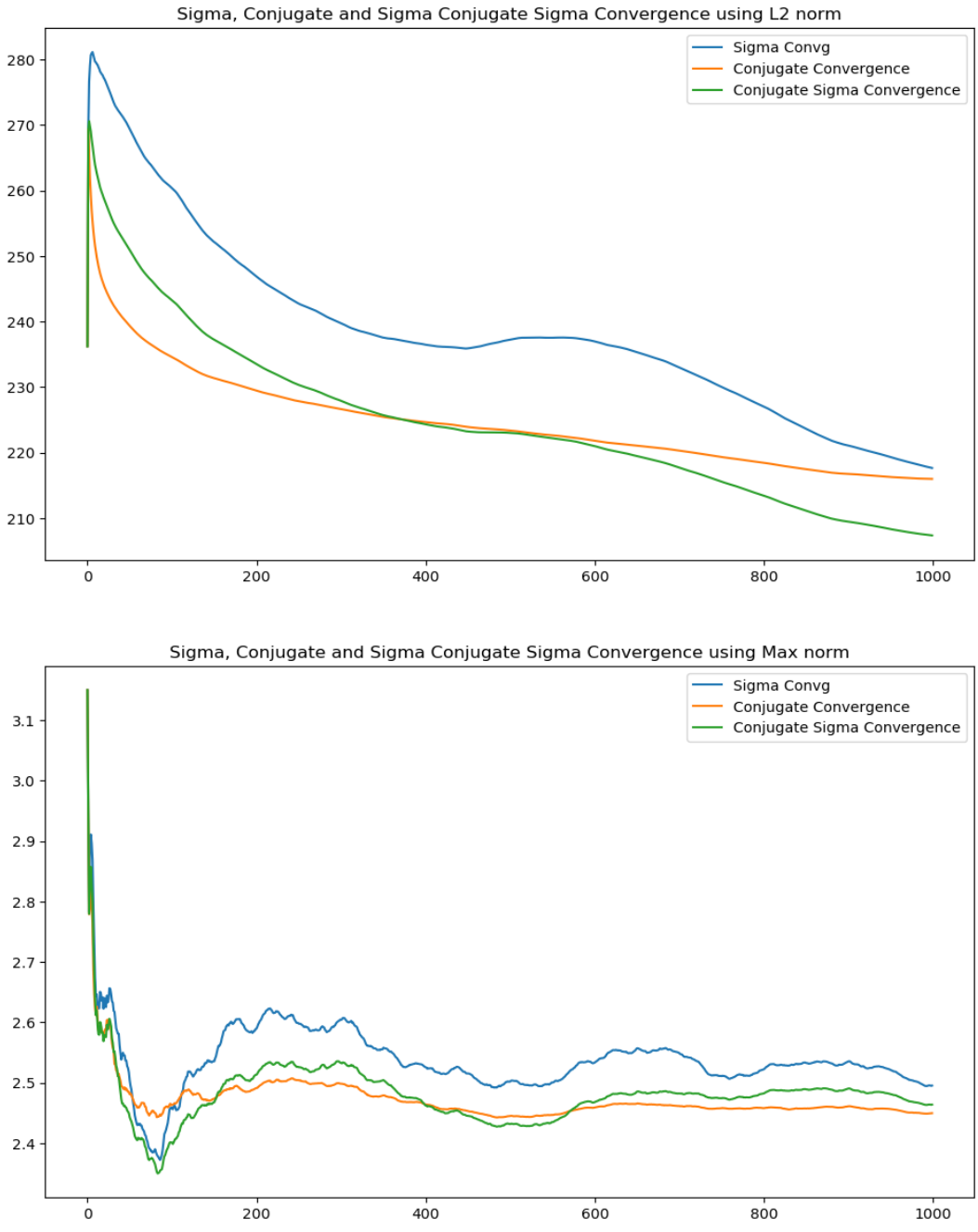


Figure 4: Comparision between Convergence techniques for 1225 dimensional matrix

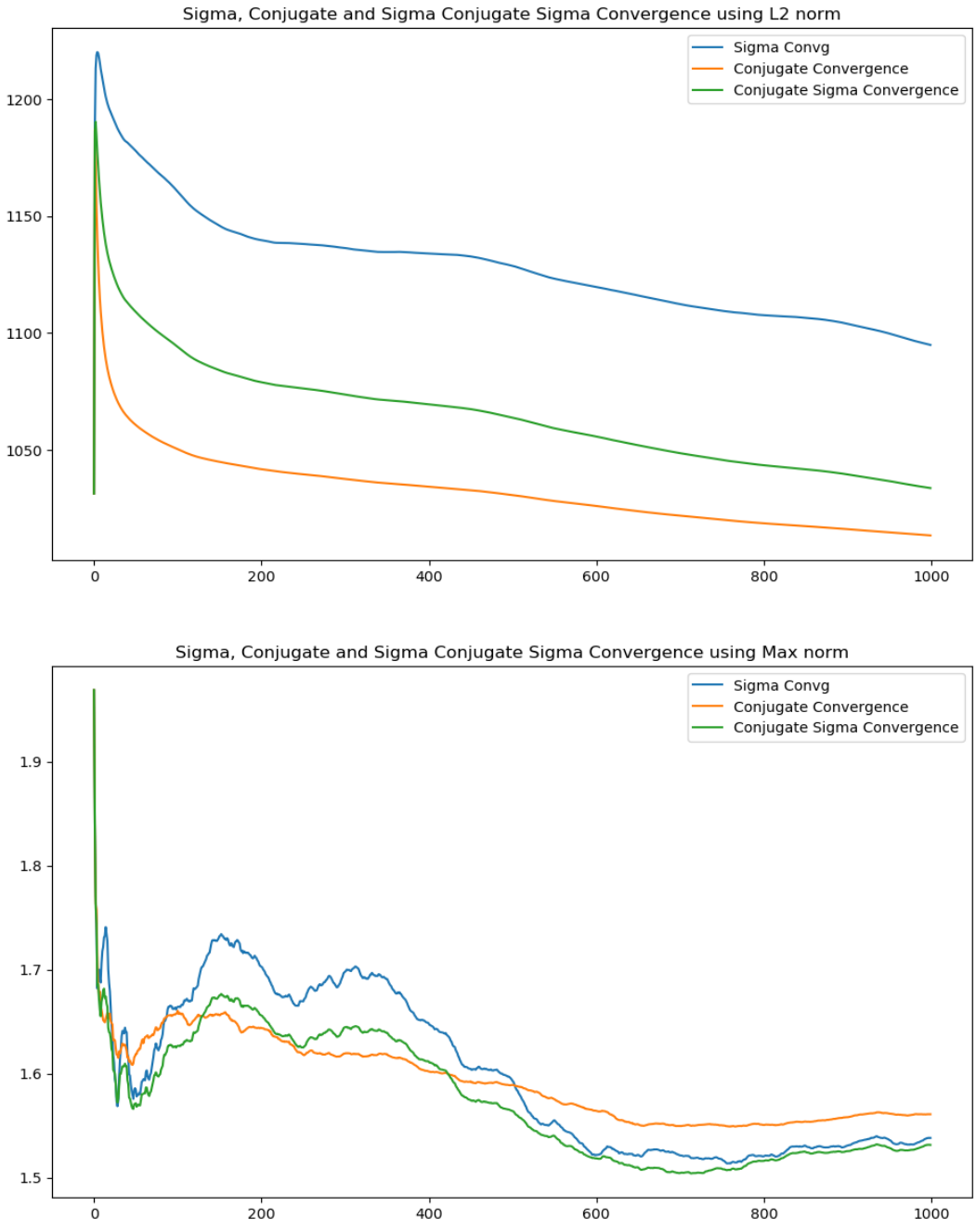


Figure 5: Comparision between Convergence techniques for 10000 dimensional matrix

3.4.6 Normal Gibbs Sampler Plots

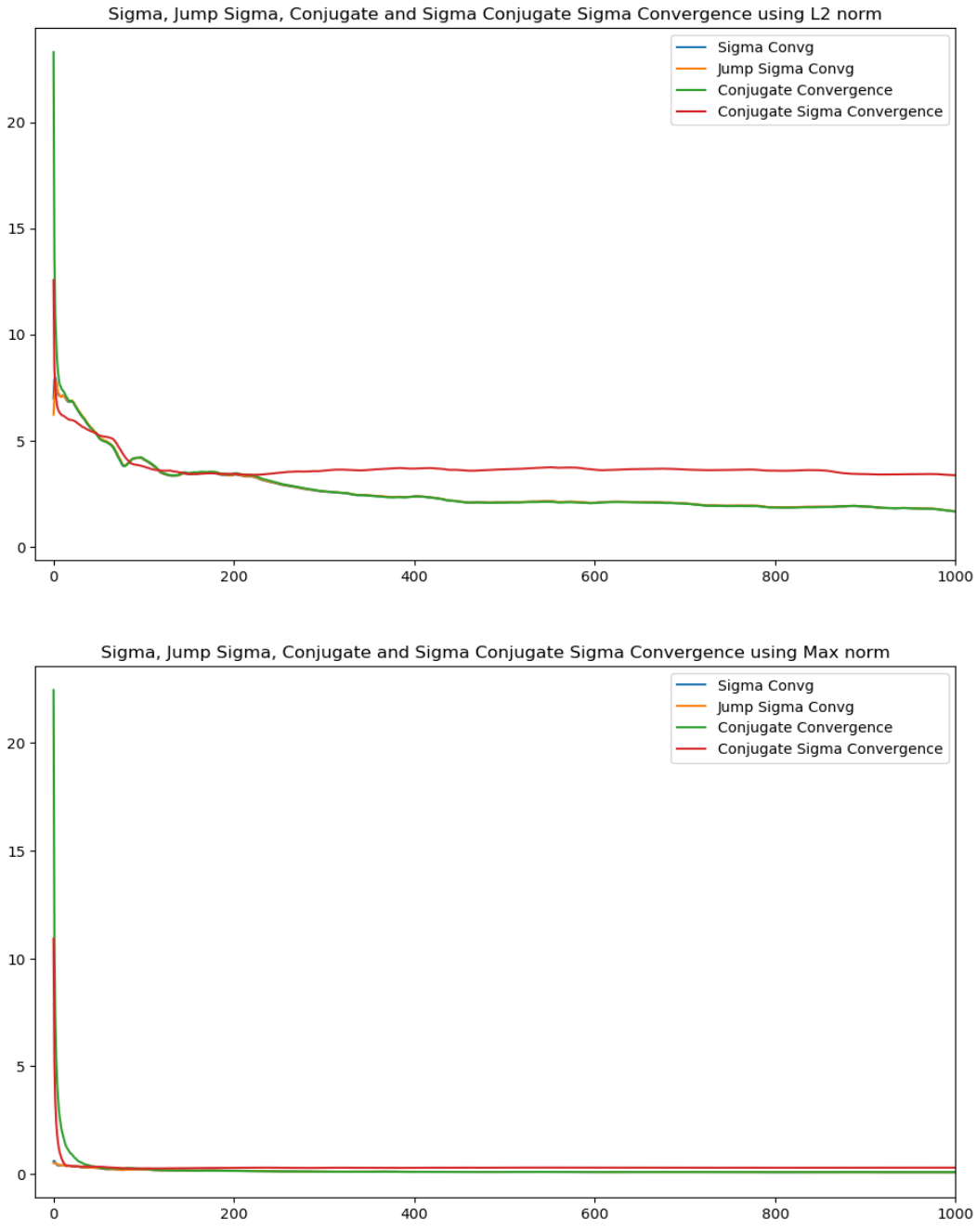


Figure 6: Comparision between Convergence techniques for 100 dimensional matrix

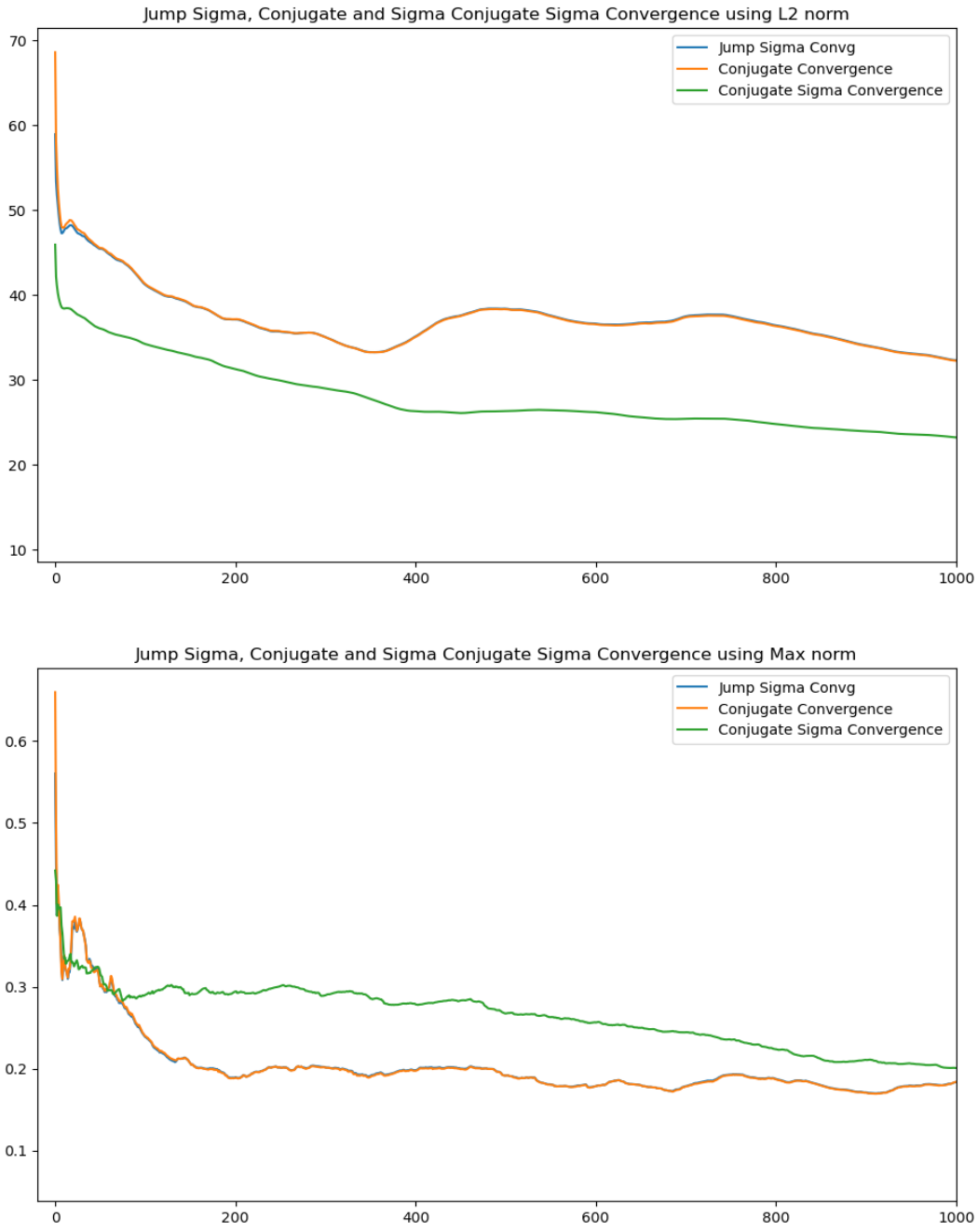


Figure 7: Comparision between Convergence techniques for 1225 dimensional matrix

Here as we can see, we aren't getting any advantage by using the Conjugate Gibbs Sampler Convergence. But Neverthless we can see that all are good approximations of A^{-1}

4 Second Component

4.1 Bayesian Network Representation of LDA Model

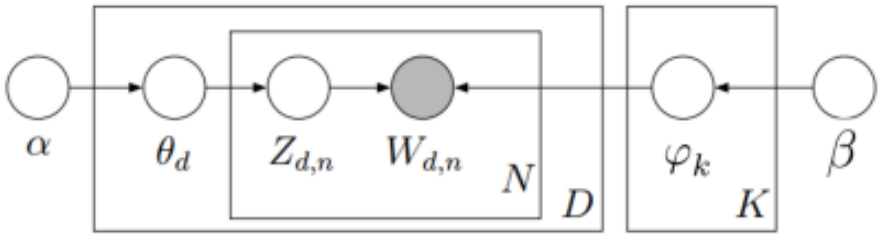


Figure 8: LDA model graphical representation

4.2 Notations

D : Number of Documents
K : Number of topics
V : Number of words in the vocabulary
N_d : Number of words in document d or Length of document d
θ_d : $< \theta_{d,t}$ for $t \in \{1, 2, ...K\} >$ Topic distribution for document d
ψ_k :$< \psi_{k,v}$ for $k \in \{1, 2, ...V\} >$Word distribution over Vocabulary V for topic k
$z_{d,n}$: Latent topic assignment to the n^{th} word in document d
$w_{d,n}$: n^{th} word in document d
Z : $\{z_{d,n}\}$, W : $\{w_{d,n}\}$, θ : $\{\theta_d\}$, ϕ : $\{\psi_k\}$
$n_{d,k,v}$: Number of {i: $w_{d,i} = v$,$Z_{d,i,k} = 1$}
$n_{d,k,:}$: $[n_{d,k,1}, ...n_{d,k,V}]$
$n_{d,k,.}$: $\sum_v n_{d,k,v}$ = Number of times Document d has topic k in it
$n_{.,k,v}$: $\sum_d n_{d,k,v}$= Number of times word v has topic k in all documents

4.3 Model

$$\psi_k \sim Dir(\beta) \text{ for } i \in \{1, 2...K\}$$
$$\theta_d \sim Dir(\alpha) \text{ for } d \in \{1, 2...D\}$$
$$z_{d,n} \mid \theta_d \sim Discrete(\theta_d)$$
$$w_{d,n} \mid z_{d,n}, \phi_{z_{d,n}} \sim Discrete(\phi_{z_{d,n}})$$

4.4 Generative Process:

⊡ "topic plate"
for all topics $k \in [1, K]$ **do**
  sample mixture components $\vec{\varphi}_k \sim \text{Dir}(\vec{\beta})$
end for
⊡ "document plate":
for all documents $m \in [1, M]$ **do**
  sample mixture proportion $\vec{\vartheta}_m \sim \text{Dir}(\vec{\alpha})$
  sample document length $N_m \sim \text{Poiss}(\xi)$
  ⊡ "word plate":
  for all words $n \in [1, N_m]$ in document m **do**
    sample topic index $z_{m,n} \sim \text{Mult}(\vec{\vartheta}_m)$
    sample term for word $w_{m,n} \sim \text{Mult}(\vec{\varphi}_{z_{m,n}})$
  end for
end for

Figure 9: Generative Model for LDA

4.5 The Need for Gibbs Sampler in LDA

Our primary task is Given Documents (W) find Z, Θ, ϕ

$$P(Z, \theta, \phi \mid W) = \frac{P(Z,\theta,\phi,W)}{P(W)}$$

P(W) can be marginalized and be written as,

$$P(W) = \int \int \int P(Z, \theta, \phi, W) \partial Z \partial\theta \partial\phi$$

$$P(W) = \frac{P(W \mid Z, \theta, \phi) \times P(Z, \theta, \phi)}{\int \int \int P(Z, \theta, \phi, W) \partial C \partial\theta \partial\phi}$$

We can elegantly factorize the above joint distribution by using our bayesian network,

$$P(Z, \theta, \phi, W) = P(\Theta \mid \alpha) P(Z \mid \Theta) P(\phi \mid \beta) P(W \mid Z, \phi)$$

$$P(Z,\theta,\phi,W) = \prod_{d=1}^{D} p(\theta_d \mid \alpha_d) \prod_{d=1}^{D}\prod_{i=1}^{N_d} p(z_{d,i} \mid \theta_d) \prod_{d=1}^{D} p(\psi_k \mid \beta_k) \prod_{d=1}^{D}\prod_{i=1}^{N_d} p(w_{d,i} \mid z_{d,i}, \phi)$$

$$P(Z,\theta,\phi,W) = \prod_{d=1}^{D} D(\theta_d;\alpha_d) \prod_{d=1}^{D}\prod_{i=1}^{N_d} (\prod_{k=1}^{K} \theta_{d,k}^{Z_{d,i,k}}) \prod_{d=1}^{D} D(\psi_k;\beta_k) \prod_{d=1}^{D}\prod_{i=1}^{N_d} (\prod_{k=1}^{K} \psi_{k,w_{d,i}}^{Z_{d,i,k}}) \tag{1}$$

We know that if $X =< x_1.....x_K >\sim Dir(\alpha_1, ...\alpha_K)$ has a pdf,

$$f(x_1, ...x_K, \alpha_1, ...\alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} (x_i)^{\alpha_i - 1}$$

where,

$$B(\alpha) = \frac{\prod_K \tau(\alpha_i)}{\tau(\sum_K \alpha_i)}$$

So, $Dir(\theta_d;\alpha_d)$ can be written as,

$$Dir(\theta_d;\alpha_d) = \frac{\prod_K \tau(\alpha_{d,k})}{\tau(\sum_K \alpha_{d,k})} \prod_{k=1}^{K} \theta_{d,k}^{\alpha_{d,k}-1}$$

In our case we assume the each dimension of the hyper parameters α and β are assumed to be equal. $\alpha =< \alpha_1, ...\alpha_K >=< \alpha, ...\alpha >$
$\beta =< \beta_1, ...\beta_V >=< \beta, ...\beta >$ If we take first two terms of equation 1 (since next to terms are mathematically analogous to first 2 terms),

$$P(\Theta \mid \alpha)P(Z \mid \Theta) = \prod_{d=1}^{D}\prod_{i=1}^{N_d} (\prod_{k=1}^{K} \theta_{d,k}^{Z_{d,i,k}}) \prod_{d=1}^{D} D(\theta_d;\alpha)$$

We can further simplify it as,

$$P(\Theta \mid \alpha)P(Z \mid \Theta) = \prod_{d=1}^{D}\prod_{k=1}^{K} (\theta_{d,k}^{\sum_{i=1}^{N_d} Z_{d,i,k}}) \prod_{d=1}^{D} \left(\frac{\prod_K \tau(\alpha_{d,k})}{\tau(\sum_K \alpha_{d,k})} \prod_{k=1}^{K} \theta_{d,k}^{\alpha-1} \right)$$

$$P(\Theta \mid \alpha)P(Z \mid \Theta) = \prod_{d=1}^{D} \frac{\prod_K \tau(\alpha)}{\tau(\sum_K \alpha)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha + n_{d,k,.} - 1}$$

Similarly we will get an analogical expression for the last two terms of equation 1 so the joint probability evaluates as,

$$P(Z,\theta,\phi,W) = \prod_{d=1}^{D} \frac{\prod_K \tau(\alpha)}{\tau(\sum_K \alpha)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha + n_{d,k,.} - 1} \prod_{k=1}^{K} \frac{\prod_V \tau(\beta)}{\tau(\sum_V \beta)} \prod_{v=1}^{V} \psi_{d,k}^{\beta + n_{.,k,v} - 1} \tag{2}$$

As we can see the above expression is for the joint probability. But while calculating the normalizing constant or the marginal factor we need to integrate the joint with respect to Z, Θ, ϕ. From the above expression it is clearly evident integrating with θ and ϕ is easy but integrating with Z is difficult because, Z is hidden inside the terms $n_{d,k,.}$ and $n_{.,k,v}$. If we assume we know Θ, ϕ then we know all the random variables that are part of the Markov Blanket of the node Z, which makes Z independent of others.

4.6 Gibbs Sampler Derivation for LDA

Let us assume we know Θ, ϕ and let's calculate probability of Z conditioned on Θ, ϕ, W.

$$P(Z \mid \Theta, \phi, W) = \frac{P(Z, \theta, \phi, W)}{\sum_Z P(Z, \theta, \phi, W)}$$

From equation 1, we can see that when we joint with evidence marginalized over Z the first and third terms of equation 1 get cancelled as they are independent of Z. So, our required probability,

$$P(Z \mid \Theta, \phi, W) = \prod_{d=1}^{D} \prod_{i=1}^{N_d} \left(\frac{\prod_{k=1}^{K} (\theta_{d,k} \psi_{k,w_{d,i}})^{Z_{d,i,k}}}{\sum_{k'} \theta_{d,k'} \psi_{k',w_{d,i}}} \right)$$

and from equation 2 we can show that,

$$P(\Theta, \phi \mid W, Z) = \frac{P(Z, \Theta, \phi, W)}{P(Z, W)}$$

from 2 we can get $P(Z, W)$ by marginalizing the joint with respect to Θ, ϕ

$$P(Z, W) = \sum_{\Theta} \sum_{\phi} P(Z, \Theta, \phi, W)$$

After integrating equation 2 with respect to Θ, ϕ,

$$P(Z, W) = \left(\frac{1}{B(\alpha)} \prod_{d=1}^{D} B(\alpha + n_{d,:,.}) \right) \left(\frac{1}{B(\alpha)} \prod_{k=1}^{K} B(\beta + n_{.,k,:}) \right)$$

which implies that

$$P(\Theta, \phi \mid W, Z) = \left(\frac{1}{B(\alpha)} \prod_{d=1}^{D} Dir(\theta_d; \alpha + n_{d,:,.}) \right) \left(\frac{1}{B(\alpha)} \prod_{k=1}^{K} Dir(\psi_k; \beta + n_{.,k,:}) \right) \tag{3}$$

From the above equation we can see the conditional independence of Θ and ϕ. We can summarize our Gibbs Sampler as follows

$$P(\Theta) \sim P(\Theta \mid Z, W, \phi)$$
$$P(\phi) \sim P(\phi \mid Z, W, \theta)$$
$$P(Z) \sim P(Z \mid \Theta, W, \phi)$$

From Equation 3 it is clearly evident that Θ and ϕ are conditionally Independent. So we can rewrite the above equations as,

$$P(\Theta) \sim P(\Theta \mid Z, W)$$
$$P(\phi) \sim P(\phi \mid Z, W)$$
$$P(Z) \sim P(Z \mid \Theta, W, \phi)$$

4.7 Collapsed Gibbs Sampler

We can see from above that Z is dependent on Θ, W, ϕ and in turn θ is dependent on Z, W and same with ϕ as well. We do not need to include, i.e., can integrate out, the parameter sets θ and ϕ because they can be interpreted as statistics of the associations between the observed $w_{m,n}$ and the corresponding $z_{m,n}$, the state variables of the Markov chain.

From $P(Z, W)$ we can derive the following collapsed expression,

$$P(Z_{d,i,k} = 1 \mid Z_{\sim d,i}, W) = \frac{(\alpha + n_{d,k,.}^{\sim d,i})(\beta + n_{.,k,w_{d,i}}^{\sim d,i})(\sum_v \beta + n_{.,k,v}^{\sim d,i})^{-1}}{(\sum_{k'} \alpha + n_{d,k',.}^{\sim d,i})(\sum_{w'} \beta + n_{.,k,w'}^{\sim d,i})(\sum_{v'} \beta + n_{.,k,v'}^{\sim d,i})^{-1}}$$

This allows to speed up the Gibbs Sampling Process because we need not touch θ and ϕ at all. To find topic assignments of a document or word distributions of a topic we can use the following equations once we obtain Z using Collapsed Gibbs Sampling. The following values are obtained by taking expectation of the samples from their respective dirchilet distribution.

$$\begin{aligned} \psi_{k,t} &= \frac{n_{.,k,v} + \beta}{\sum_{v=1}^{V}(n_{.,k,v} + \beta)} \\ \theta_{d,k} &= \frac{n_{d,k,.} + \alpha}{\sum_{k=1}^{K}(n_{d,k,.} + \alpha)} \end{aligned} \tag{4}$$

4.8 PseudoCode for the Sampler

⊡ initialisation
zero all count variables, $n_m^{(k)}, n_m, n_k^{(t)}, n_k$
for all documents $m \in [1, M]$ **do**
  for all words $n \in [1, N_m]$ in document m **do**
    sample topic index $z_{m,n}=k \sim \text{Mult}(1/K)$
    increment document–topic count: $n_m^{(k)} + 1$
    increment document–topic sum: $n_m + 1$
    increment topic–term count: $n_k^{(t)} + 1$
    increment topic–term sum: $n_k + 1$
  end for
end for
⊡ Gibbs sampling over burn-in period and sampling period
while not finished **do**
  for all documents $m \in [1, M]$ **do**
    for all words $n \in [1, N_m]$ in document m **do**
      ⊡ for the current assignment of k to a term t for word $w_{m,n}$:
      decrement counts and sums: $n_m^{(k)} - 1; n_m - 1; n_k^{(t)} - 1; n_k - 1$
      ⊡ multinomial sampling acc. to Eq. 79 (decrements from previous step):
      sample topic index $\tilde{k} \sim p(z_i | \vec{z}_{-i}, \vec{w})$
      ⊡ use the new assignment of $z_{m,n}$ to the term t for word $w_{m,n}$ to:
      increment counts and sums: $n_m^{(\tilde{k})} + 1; n_m + 1; n_{\tilde{k}}^{(t)} + 1; n_{\tilde{k}} + 1$
    end for
  end for
  ⊡ check convergence and read out parameters
  if converged and L sampling iterations since last read out **then**
    ⊡ the different parameters read outs are averaged.
    read out parameter set $\underline{\Phi}$ according to Eq. 82
    read out parameter set $\underline{\Theta}$ according to Eq. 83
  end if
end while

Figure 10: Pseudo code for the collapsed sampler

4.9 Experiments

4.9.1 DataSet Description

I took **fetch20newsgroups** dataset which was available on sklearn library. I did data pre-processing steps like Stemming (removing common stop words like 'is', 'the') and vectorized the words. Numerical Details of the dataset are mentioned below:

Number of Documents = 1000
Number of words in the Vocabulary = 7358
Number of topics = 10 (can be varied in the code)

4.9.2 Hyperparameters

$\alpha = 50/\mathrm{T}$ where T is number of topics
$\beta = 0.01$

4.9.3 Comparision between results obtained by Normal Gibbs Sampler and Conjugate Gibbs Sampler

Sampler	Number of iterations	Training Time(in sec)
Gibbs Sampler	500	1137
Collapsed Gibbs Sampler	500	830

Table 1: Training times

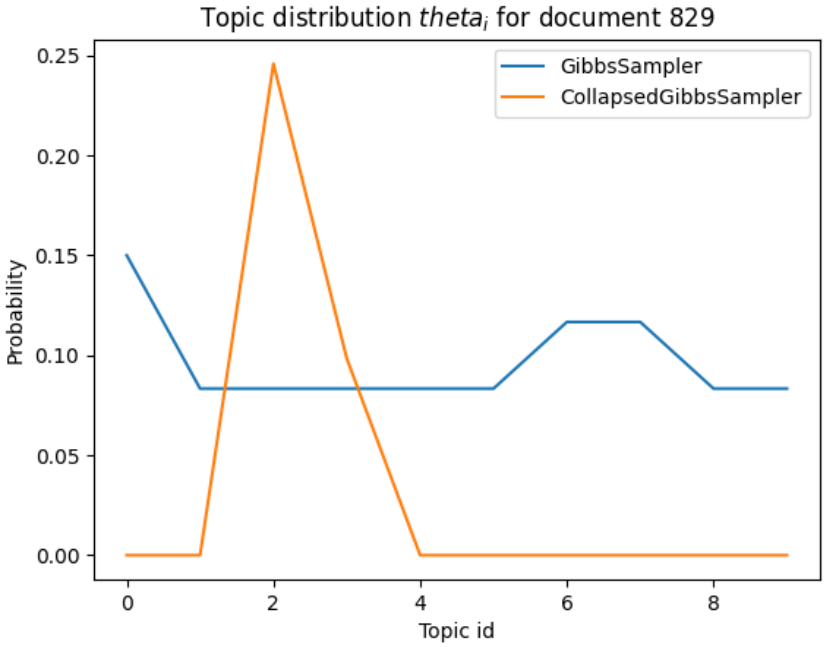


Figure 11: Theta distribution inferred using Gibbs Sampler and Collapsed Gibbs Sampler for document 829

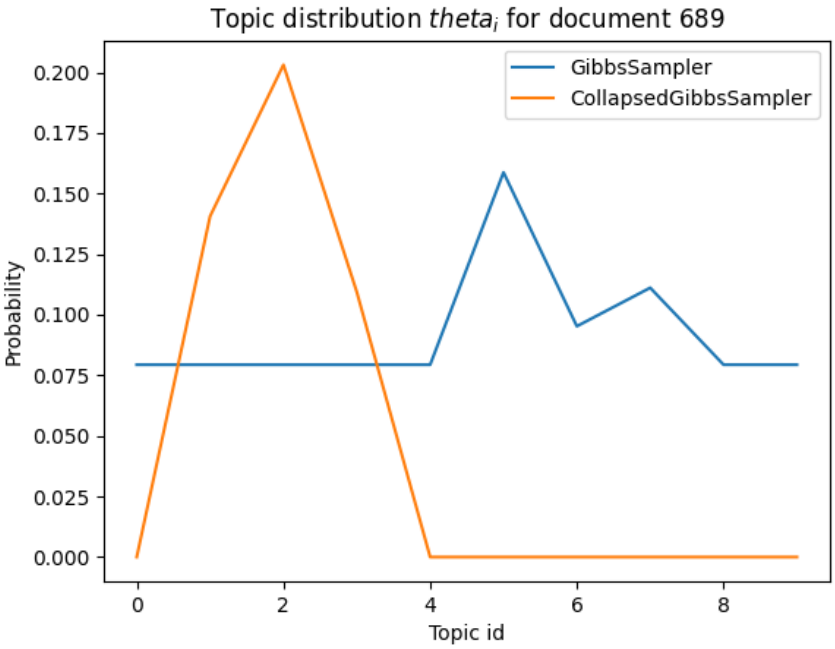


Figure 12: Theta distribution inferred using Gibbs Sampler and Collapsed Gibbs Sampler for document 689

```
Topic #0: com space new available data 20 information use 10 aids
Topic #1: just know like people don year years car use time
Topic #2: server support edu supported os joseph david file readme vga
Topic #3: edu graphics pub mail ray send ftp objects server files
Topic #4: section firearm license military shall weapon person following means u
se
Topic #5: like know think problem windows use don just good does
Topic #6: like just key don good government people think encryption use
Topic #7: people god just don think like time good know israel
Topic #8: edu navy vote votes health mil misc car hp thomas
Topic #9: goal game finnish shot puck roger peter sweden good slave
```

Figure 13: Topics with their top 10 words from the vocabulary inferred by Gibbs Sampler

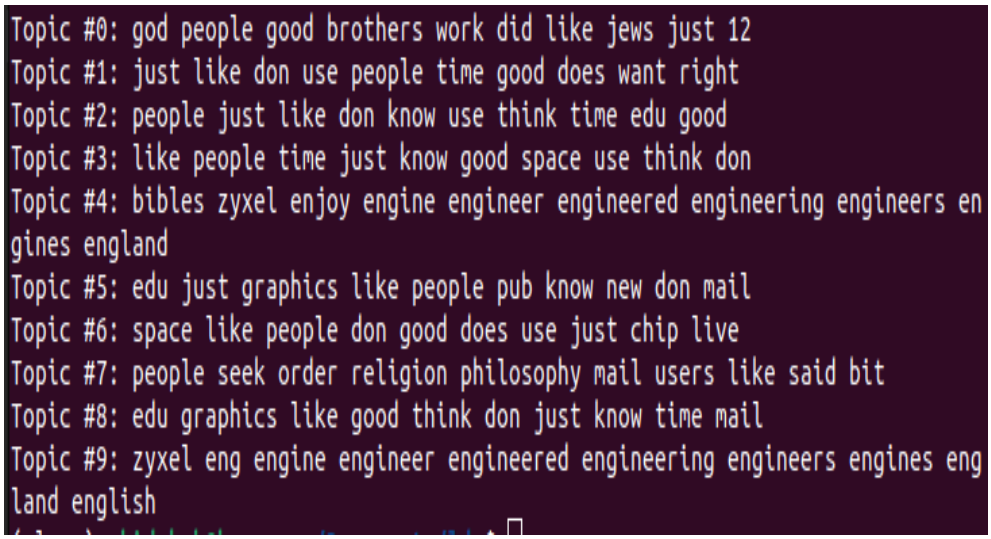


Figure 14: Topics with their top 10 words from the vocabulary inferred by Collapsed Gibbs Sampler

4.10 Observations

- As we can clearly see from the table Collapsed Gibbs Sampler is working faster than Normal Gibbs Sampler.
- From the top 10 words of each topic it looks like Gibbs Sampler did a better job in classifying the words into respective topics than Collapsed Gibbs Sampler.

4.11 Analysing our above topic Model for querying

After Training the Model we have θ, ϕ, Z. How to assign topic distributions for a given new Document D' which was not seen in the training data? Assuming the new document contains words which are already part of our Vocabulary. It doesn't contain any new words.

Before solving the above task we will first formulate what we have and what we need. We need,

$$P(Z, \theta \mid W, \phi)$$

Since the words are already in our Vocabulary W and ϕ won't change. Even in this case as Z changes we can't marginalize our total joint with respect to Z. Hence, we take help of our Collapsed Gibbs sampler again. We can sample Z using collapsed Gibbs Sampling and when we are confident enough on our Sampler we sample θ according to the equations 4.

References

[1] F Naeimi Dafchahi. A new refinement of jacobi method for solution of linear system equations ax= b. *Int. J. Contemp. Math. Sciences*, 3(17):819–827, 2008.

[2] Mehdi Dehghan, Behnam Hashemi, and Mehdi Ghatee. Computational methods for solving fully fuzzy linear systems. *Applied mathematics and computation*, 179(1):328–343, 2006.

[3] Nicholas J Higham. Cholesky factorization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):251–254, 2009.

[4] Oleksandra Osadcha and Zbigniew Marszaek. Comparison of steepest descent method and conjugate gradient method. In *CEUR Workshop Proceedings, System*, 2017.

[5] Ivan V Oseledets and Sergey V Dolgov. Solution of linear systems and matrix inversion in the tt-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

[6] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[7] Wikipedia contributors. Schur complement — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Schur_complement&oldid=1041999862`, 2021. [Online; accessed 21-April-2022].