ANALYSIS OF A GIBBS SAMPLING ALGORITHM FOR NON-NEGATIVE MATRIX FACTORIZATION

ACKNOWLEDGEMENT

I am very grateful to my project supervisor Dr Ivo Siekmann for his excellent guidance and support in completing the project. I'm sincerely thankful for his help and transparency during my research. His inputs and advice have helped me a lot by doing this thesis—also, many thanks to Dr Ivan Olier-Caparroso for the dataset used in the project. I'd also like to express my gratitude to all of my friends who assisted me with valuable project feedback and support in completing the project report. I am heartily thankful to Liverpool John Moores University for allowing me to do this project and providing me with the necessary resources needed to complete the project. Every minute I spent working on this project has aided my knowledge and skills development.

Any future recommendations for improving the project would be greatly appreciated and gratefully recognised.

TABLE OF CONTENTS

ABSTRACT: Non-negative matrix factorization (NMF) is an algorithm based on the factorization of a non-negative data matrix $\boldsymbol{X}$ in two non-negative factors, $\boldsymbol{A}$ and $\boldsymbol{S}$, such that $\boldsymbol{X} = \boldsymbol{AS}$. It allows for unsupervised separation of source signals and is used for various purposes such as dimensionality reduction, feature selection, classification, and deconvolution of data generated by mixing several sources. The project's main aim was the development of the NMF based on Moussaoui et al. (2006) and then investigating this algorithm's behaviour. There were a lot of challenges while translating the equation to code, as the paper contained various mistakes in the equations, which were corrected after certain research. During this project, Gibbs sampling was constructed for sampling the signals. The performance of the Gibbs sampler was accessed using methods like convergence plots which showed that the convergence of the sampler was very slow and hence it is an inefficient sampler. The project mainly revolves around the equation $\boldsymbol{X} = \boldsymbol{AS} + \boldsymbol{E}$, where $\boldsymbol{A}$, $\boldsymbol{S}$, and $\boldsymbol{E}$ are mixing coefficients, source signals, and noise signals. NMF in a Bayesian framework has been discussed in the document. Also, the Markov chain Monte Carlo method has been derived to estimate their posterior density based on a Gibbs sampling procedure. For the creation of non-negative sources, gamma distribution was used. A Markov chain Monte Carlo (MCMC) sampling process is provided to simulate the resultant joint posterior density, from which marginal posterior mean estimates of the source signals and mixing coefficients are derived. This document contains the proper explanation of the methods followed to accomplish the project.

1. INTRODUCTION

Non-negative matrix factorization is an algorithm that allows for unsupervised separation of source signals, also known as blind source separation (Lopes & Ribeiro, 2015, Moussaoui et al. (2006)). The algorithm has been used for various tasks like dimensionality reduction, feature selection, classification, deconvolution of data generated by mixing several sources (Lopes & Ribeiro, 2015, Moussaoui et al. (2006)). The method is based on factorization of a non-negative data matrix $\boldsymbol{X}$ in two non-negative factors $\boldsymbol{A}$ and $\boldsymbol{S}$ such that $\boldsymbol{X} = \boldsymbol{AS}$ Moussaoui et al. (2006). NMF has recently received much attention as unsupervised learning due to its effective application to the classification and clustering task (Colyer, 2019). The limitation of non-negativity is normal for most natural signals like amplitude spectrums, pixel intensities, and occurrence counts. NMF has been widely used in environmental metrics and chemometrics to find the explanatory source signals in a series of chemical mixtures (Duarte, L et.al 2014). It is also used in image processing

to find useful features (Colyer, 2019). NMF also finds its use in text processing and audio processing (Colyer, 2019).

In any case, a Bayesian methodology depends on prior distributions of matrices and posteriors of the observed data. The conventional Bayesian NMF approach estimates posterior density using Gibbs sampling and ensures non-negativity by using exponential priors (Schmidt et al., 2009). This gives the marginal posterior density, which helps compute uncertainty estimates (Schmidt et al., 2009). The report's main topic is to investigate a Bayesian approach to NMF and develop a Markov Chain Monte Carlo (MCMC) algorithm for sampling from the posterior distribution (Taboga, 2021, Moussaoui et al. (2006)**)**. In the field of statistics, MCMC methods are a class of algorithms that provides approaches to a random sampling of high-dimensional probability distributions given a condition where the next sample is dependent on the state of the current sample (Agrahari, 2021). The whole project was based on developing the algorithm based on Moussaoui et al. (2006) which addresses the blind-source separation in cases where both the source signals and their mixing coefficients are non-negative. The paper uses spectral data and expresses the linear mixing model as

$$x_t = \boldsymbol{A}s_t + e_t \text{ for} t = 1,2 \ldots., N \quad (1)$$

Where, s_t represents source signals of dimension $(n \times 1)$, x_t represents measured signals of dimension (m×1) and e_t represents the noise signals of $(m \times 1)$ dimension and $\boldsymbol{A}$ represents $(m \times n)$ unknown mixing matrix containing the mixing coefficients and t is the index of the N observed variables. The above equation (1) will be referred to as $\boldsymbol{X} = \boldsymbol{AS} + \boldsymbol{E}$ throughout the report. The matrices $\boldsymbol{A}, \boldsymbol{S}$ and $\boldsymbol{E}$ are sometimes referred to as mixing coefficients, source signals and noise signals respectively.

The document mainly discusses the NMF in a Bayesian framework. NMF is just a matrix decomposition i.e., factorization of a matrix $\boldsymbol{X} = \boldsymbol{AS}$ where the number of columns or the number of columns of $\boldsymbol{A}$ is a parameter of the method, maximum likelihood or maximum posterior estimator are used for finding solutions and under the condition that the data matrix $\boldsymbol{A}$ is not exactly known but perturbed by noise Moussaoui et al. (2006). Based on a Gibbs sampling approach, the Markov chain Monte Carlo method was developed to determine their posterior density. The posterior density can be derived using the posterior density estimations using Markov Chain Monte Carlo (including maximum likelihood estimators). The study also includes a discussion on how the procedures might be improved in the future.

The project's main objective was to implement the algorithm on NMF by Moussaoui et al. (2006) and then investigate the behaviour of this algorithm, in particular, revealing non-identifiability. The benefits of this implementation are multiple. The advantages of NMF are Sparsity is enforced through non-negativity (Moussaoui et al. (2006), Peharz & Pernkopf, 2012). Non-negative models are always zero when a given element does not contribute to a signal, but unconstrained decompositions are nearly never zero Moussaoui et al. (2006). When we wish to uncover separate feature sets or sample associations, we desire sparse representations. Non-negativity prevents elements from cancelling one other out (Ke-Lin & Swamy, 2019). For example, if one element "overcorrects" a signal, another factor may try to "counter-correct" to bring the signal back into balance (Ke-Lin & Swamy, 2019). Factors that are exclusively positive or zero can never be counter corrected and can only explain additive signals. An unconstrained orthogonal factorization will almost always be non-negative in a well-conditioned non-negative input matrix. Imposing non-negativity, on the other hand, imposes appropriate theoretical assumptions and can speed up convergence. Imputation of missing signals is encouraged by non-negativity (Schmidt et al., 2009). Because signals can only be interpreted in one way—additively—much more missing signal is imputed than in SVD (Schmidt et al., 2009). If negative values are included in the models, the algorithm may be "tempted" to delete genuine data points instead of adding to missing ones (Schmidt et al., 2009). Because data in the matrix to be factored is frequently partial owing to signal dropout or inadequate sampling, imputation is critical. The algorithm was implemented in the R programming language (R Core Team 2020) using RStudio (Version 2021.09.0+351 / Windows).

LITERATURE REVIEW

Lots of research and implementations have been done for the NMF and its use cases. The idea of the NMF can be traced back to 1994 in a paper (Paatero, P. & Tapper, U. 1994) by Paatero and Tapper before Lee and Seung popularised it in paper (Lee, D.D. & Seung, H.S. 1999). The experiment's main objective was to perform factor analysis on environmental data, Factor analysis is a method for condensing a large number of variables into a smaller number of components. This method takes the largest common variance from all variables and converts it to a single score (Lee, D.D. & Seung, H.S. 1999). Each factor was a positive combination of sources. In 1999, Lee and Seung used NMF in a paper (Lee, D.D. & Seung, H.S.1999) on unsupervised learning. In the paper,

the authors compared NMF with the Principal Component Analysis (PCA) and Vector Quantization (VQ). The authors applied NMF together with Principal component analysis and Vector quantization to facial images in a database in which all the methods learn to represent a face as a linear combination of the basic images. Still, the results were qualitatively different. The paper suggests that each of PCA, VQ in their own way. But the interesting feature of NMF is that it detects components in a set of similar images. The paper explains how NMF learns representations of images that are different from that of PCA and VQ by describing the three methods in a matrix factorization framework.

In another paper (Schmidt, M.N. et.al 2009), the authors implemented the procedure for Bayesian treatment of non-negative matrix factorization which was particularly based on exponential priors and normal likelihood and derived a proper Gibbs sampler to approximate the posterior density of the NMF factors. The whole implementation was carried out on a chemical brain imaging database. The authors showed the method to be valuable for interpretability of chemical brain imaging data set by providing uncertainty estimates. The paper further discusses the use of Gibbs sampler for model order selection by estimating the marginal likelihood and compares it with the Bayesian information criterion. An iterated conditional modes method was used for computing the maximum posterior estimate.

The normal distribution is the most common model of randomness. One drawback of the normal distribution is that it provides a positive probability density to every value in the range $(-\infty, +\infty)$. But in some of the cases (this project was one of them), it is desired to use the normal distribution to describe the random variations of the strictly positive quantities.

Here, a proper way to preserve the main features of the normal distribution while avoiding negative values involves the truncated normal distribution, in which the range of definition is made finite at one or both ends of the interval. A truncated distribution is obtained from the normal distribution with its domain restricted to a specific range of values. The need for simulation of truncated normal variables was discussed in the paper 'simulation of truncated normal variables in which the authors stated that the need for truncated normal simulation appears in Bayesian inference for some truncated parameter space problems.

In the paper (Zhong & Girolami, 2009), the authors presented a Bayesian approach to NMF by developing a Reversible Jump MCMC method that provided full posteriors over the components of the given matrix. The authors also addressed the issue with NMF model selection (i.e.,

estimating the number of rows of the source matrix in the equation). According to the authors, the adopted procedure provided the posterior distribution over the matrix dimensions, thereby providing the number of components in the NMF model. The paper also provided the comparative analysis of Bayesian Information Criterion and model selection employing marginal likelihood estimates. The authors investigated the power and flexibility of the Bayesian methodology and RJMCMC procedure. They assessed differing model structures, and through the findings, the author inferred the corresponding plausible component spectra for the complex data.

The whole methodology of this project is based on a paper by Moussaoui et al. (2006), which addresses blind source separation in case both source signal and mixing coefficients are positive. The application revolved around the analysis of spectrometric data sets. The non-negative source separation is performed in a Bayesian framework in which non-negativity was encoded through the assignments of Gamma priors on the distribution of source as well as mixing coefficients. The authors proposed an MCMC sampling procedure to simulate the resulting joint posterior density from which marginal posterior mean estimates of signals are gathered. With the help of the outputs obtained from synthetic and experimental spectra are used to discuss the problem of non-negative source separation.

2. DESCRIPTION OF DATASET

Data is the basis of every study and research. Since the project was of method development, the data were mainly used to test the validity of the methods. So, the first step in this project was developing the dataset before developing the methods.

2.1.Dataset Generation from Gamma Distribution

Firstly, a set of R functions was used for generating the test datasets consistent with the statistical model used for the development of the algorithm. Different matrices were generated as a test dataset using R code. For generating the matrices with non-negative factors, sampling from the gamma distribution seemed suitable. Thus, for generating the dataset gamma distribution was used for the source matrix $\boldsymbol{S}$ and mixing coefficient $\boldsymbol{A}$. The reason behind generating the gamma-distributed dataset was the requirement to produce non-negative variables as gamma distribution is a continuous probability distribution used to model continuous variables that are always positive and have skewed distributions (Sengupta, 2020). After that, the errors were sampled from normal distribution and were added to the product of $\boldsymbol{A}$ and $\boldsymbol{S}$. As a result, we had "true" matrices $\boldsymbol{A}$ and $\boldsymbol{S}$ and a data matrix $\boldsymbol{X}$ formed after adding errors to $\boldsymbol{AS}$.

The function generateTestData from the code can generate m samples from n sources with N variables. The matrix generated were consistent with the noise level σ. The dataset generated looked like the following matrices.

$$\boldsymbol{X} = \begin{pmatrix} 4004904 & 1399595.9 \\ 1438149 & 413688.4 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 1801.7213 & 1039.2402 \\ 230.8338 & 828.7767 \end{pmatrix}$$

$$\boldsymbol{S} = \begin{pmatrix} 1455.789 & 582.4717 \\ 1329.795 & 336.9234 \end{pmatrix}$$

Test dataset generated using custom dataset generation function

At the beginning of the project, these matrices with small dimensions were used for faster processing and assuring the correctness of the methods developed. The same functions were later used to produce the matrices with higher dimensions and used the matrices to test the developed methods.

2.2.Dataset by Linear mixing of Images

For further testing of the developed methods using more sophisticated dataset, the next set of testing dataset which was generated using the linear mixing of images was taken. The dataset was the courtesy of Dr Ivan Olier-Caparroso. The dataset containing 2000 images each with the dimension of 50×50 was formed by mixing four different images provided to me. Thus, the number of sources was four for the dataset. These sources were linearly mixed. The source image matrix was denoted by $\boldsymbol{S}$, coefficient matrices were denoted by $\boldsymbol{A}$ and noise were denoted Epsilon (ε). The whole dataset matrix was denoted by $\boldsymbol{X}$ which was the linear combination of source $\boldsymbol{S}$, coefficient $\boldsymbol{A}$ and noise matrix σ. Following were some sample images from the dataset.

Figure 1: Sample Images from the test dataset

As from the images it can be seen traces of Rose, Zebra, Eiffel Tower and Mountain. The distinct image close to the original can be retrieved from the shown images simply by the matrix factorization.

3. METHODOLOGIES

The algorithm was developed in R studio using the R programming language. For the algorithm, the paper by Moussaoui et al. (2006) was taken as the reference. The steps in algorithm development involved investigating Bayesian non-negative matrix factorization and using Gibbs Sampler (a specific MCMC algorithm) for non-negative matrix factorization to sample source $\boldsymbol{S}$, mixing coefficients $\boldsymbol{A}$ and noise. The steps are explained below:

3.1.Investigation of Bayesian Modelling

The NMF problem can be modeled as $\boldsymbol{X} = \boldsymbol{AS} + \boldsymbol{E}$, where $X \in R^{Im \times Jn}$ is a data matrix which can be factorized as the product of two element-wise non-negative matrices, $A \in R^{Im \times Nn}$ and $S \in R^{Nn \times JN}$ up to an additive error $E \in R^{Im \times JN}$.

In the Bayesian method, the knowledge of distribution of the residual can be stated in terms of a likelihood function, and the parameters in terms of prior densities. The priors are so chosen in accordance with the beliefs about the distribution of the parameters; however, to allow efficient inference in the model it is desirable to choose prior densities with a convenient parametric form. The idea behind Bayesian approach to source separation is to make use of both likelihood and prior information on the source signals and the mixing process through the assignment of prior distributions $p(S)$ and $p(A)$. According to the Bayes' theorem, the joint posterior density can be written as

$$p(X) = p(S, A) \times p(S) \times p(A) \div p(X) \tag{1}$$

where, $\boldsymbol{A}$ and $\boldsymbol{S}$ are assumed to be independent of each other. Since $p(X)$ it is a normalization constant the equation can be written as.

$$p(X) \propto p(S, A) \times p(S) \times p(A) \tag{2}$$

Prior, in Bayesian statistical inference, is the probability of certain events before new evidence is collected. Similarly, likelihood is simply the chance of an event to occur. Here for this case, $p(X|S, A)$ is the likelihood and $p(S)$ and $p(A)$ are priors. From this posterior density, both $\boldsymbol{A}$ and

$\boldsymbol{S}$ can be estimated by using various Bayesian estimators. However, the initial step in inference is to use appropriate probability distributions to encode prior information about noise sequences, source signals, and mixing coefficients.

3.1.1. Derivation of the Likelihood

Before encoding the noise sequences, the sequences are assumed to be independent and identically distributed. Thus, the likelihood of error are modelled using the defined prior as:

$$p(\theta_1) = \prod_{t=1}^{N} \prod_{i=1}^{m} \mathcal{N}\left(\mathrm{e}_{(\mathrm{i,t})}; 0, \sigma_i^2\right) \tag{3}$$

Where, $\mathcal{N}\,(\mathrm{z}; \mu, \sigma^2)$ is the normal distribution, and $\theta_1 = \{\,\sigma_i^2\,\}_{i=1}^{m}$

Likelihood can be calculated by the equation.

$$p(X, S, \theta_1) = \prod_{t=1}^{N} \prod_{i=1}^{m} \mathcal{N}\left(\mathrm{x}_{(\mathrm{i,t})}; \sum_{k=1}^{n} a_{(i,k)} s_{(k.t)}\,, \sigma_i^2\right) \tag{4}$$

3.1.2. Source Signals and Mixing Coefficients Priors

The sources are considered to be statistically independent of one another, with each jth source signal being i.i.d. and having a Gamma distribution of parameters (α_j, β_j) . The associated prior densities are then expressed by

$$p(\theta_2) = \prod_{t=1}^{N} \prod_{i=1}^{m} \mathcal{G}\left(s_{(j.t)}; \alpha_j, \beta_j\right) \tag{5}$$

$$p(\theta_3) = i = 1Ni = 1m\mathcal{G}\left(a_{(i,j)}; \gamma_j, \lambda_j\right) \tag{6}$$

Where the vectors $\theta_2 = \left\{\alpha_j, \beta_j\,\right\}_{j=1}^{n}$ and $\theta_3 = \left\{\gamma_j, \lambda_j\,\right\}_{j=1}^{n}$. These contains the parameters of the gamma distributions

3.2.Gibbs sampling

Gibbs sampling (S.Geman & D.Geman, 1984) is a specific type of Markov chain Monte Carlo algorithm that approximates a complex probability distribution by sampling from a specified multivariate probability distribution. It is usually used when the joint distribution is difficult to sample directly, but the conditional distribution is known for each variable and is easier to sample from. In this sampling method, a sequence of samples are drawn from the conditional posterior densities of the parameters which converge to a sample from the joint posterior (Zavaleta et.al 2009). Then source signals $\boldsymbol{S}$ and mixing coefficients $\boldsymbol{A}$ are calculated. Here, θ is the hyperparameter. For this particular case, there were a total of five hyperparameters.

The steps followed for method development are described below:

Case 1: Assuming all hyperparameters are known

At first, the methods were implemented assuming that the noise level σ and the parameter distribution was known. This helped in simplifying the code and helped for easy testing of the algorithm.

Sampling the Source $\boldsymbol{S}$

For implementing the Gibbs sampler first, the source sampler was made. The first step in sampling the Source began with modeling the code for the equation (37) in paper Moussaoui et al. (2006). The equations can be seen below:

$$p(x_{(1:n,t)}|s_{(j,t)}, \theta_{s(j,t)}^{likel}) \propto \exp\left\{-\frac{\left(s_{(j,t)} - \mu_{s(j,t)}^{likel}\right)^2}{2[\sigma_{sj}^{likel}]^2}\right\} \quad (7)$$

Where, σ_{sj}^{likel} and $\mu_{s(j,t)}^{likel}$ are given by:

$$\left[\sigma_{sj}^{likel}\right]^2 = \left[\Sigma_{i=1}^{m} \frac{\left[a_{(i,j)}^{(r)}\right]^2}{\left[\sigma_i^{(r)}\right]^2}\right]^{-1} \quad (8)$$

$$\mu_{s(j,t)}^{likel} = [\sigma_{sj}^{likel}]^2 \Sigma_{i=1}^{m} \frac{\left(a_{(i,j)}^{(r)}\ \varepsilon_{(i,t)}^{(-j)}\right)}{[\sigma_i^{(r)}]^2} \tag{9}$$

Here r in the above equations represents the current iteration of Gibbs sampler.

This is the example, from $\sigma_s(j,.)$ we can calculate $\mu_{s(j,t)}^{like}$. A mistake from Moussaoui et al. (2006) must be fixed. In paper the formula was,

$$\mu_{s(j,t)}^{likel} = \frac{1}{\left[\sigma_{s\,j}^{likel}\right]^2} \sum_{i\,=\,1}^{m} \frac{a_{(i,j)}^{(r)}\ \varepsilon_{(i,t)}^{(-j)}}{[\sigma_i^{(r)}]^2}$$

But the simplified formula must be,

$$\mu = [\sigma_{s(j,.)}^{like}]^2 \sum_{i=1}^{m} \frac{a_{(i,j)}^{(r)}\ \varepsilon_{(i,t)}^{(-j)}}{\sigma_i^2}$$

The error (ε) due to the omission of a jth row of $\boldsymbol{S}$ and jth column of $\boldsymbol{A}$. was calculated using the following equation.

$$\varepsilon_{(i,t)}^{(-j)} = x_{(i,j)} - \Sigma_{k=1}^{j-1} a_{(i,k)}^{(r)} s_{(k,t)}^{(r+1)} - \Sigma_{k=j+1}^{n} a_{(i,k)}^{(r)} s_{(k,t)}^{(r)} \tag{10}$$

From the codes for equation $(9), (10)$ the source matrix $\boldsymbol{S}$ was sampled using the folded normal distribution and variance was calculated from equation (8) of the paper.

Sampling $\boldsymbol{A}$

For this, the error distribution of columns of $\boldsymbol{A}$ were calculated from source matrix $\boldsymbol{S}$ and Standard deviation of errors (σ) was also calculated using the equation (45) of the paper Moussaoui et al. (2006).

$$p(x_{(i,1:N)} \big|\ a_{\ (i,j)} \theta_{a(i,j)}^{likel}) \propto \exp\left\{ - \frac{(a_{(i,j)} - \mu_{a(i,j)}^{likel})^2}{2\left(\sigma_{a(i,j)}^{likel}\right)^2} \right\} \tag{11}$$

Where,

$$\sigma_{a(i,j)}^{likel}]^2 = \frac{[\sigma_i^{(r)}]^2}{\sum_{t=1}^{N}[s_{jt}^{(r+1)}]^2} \tag{12}$$

$$\mu_{a(i,j)}^{likel} = [\sigma_{a(i,j)}^{likel}]^2 \sum_{t=1}^{N} \frac{s_{jt}^{(r+1)} \varepsilon_{it}^{(-j)}}{[\sigma_i^{(r)}]^2} \tag{13}$$

For $\boldsymbol{A}$, both equations (6) and (7) were modelled in R. For **A**, the deviation of and (i.e., ε (Epsilon)) code was written. The sampling code was written similarly to that of the sampling $\boldsymbol{S}$ code. Firstly, the error of columns of $\boldsymbol{A}$ from source matrix $\boldsymbol{S}$ and standard deviation of errors σ using equation (8) shown above. Also, the new matrix $\boldsymbol{S}$, which was sampled earlier using the steps mentioned in Section Sampling $\boldsymbol{S}$ instead of the initially generated $\boldsymbol{S}$. The deviation of $\boldsymbol{X}$ and $\boldsymbol{A} \cdot \boldsymbol{S}$ was calculated again with the condition when source j is omitted. Finally, after calculating σ_{sj}^{likel} and $\mu_{s(j,t)}^{likel}$, matrix $\boldsymbol{A}$ was sampled from a folded normal distribution. The random variable $Y = |X|$ has a folded normal distribution when given a normally distributed random variable $\boldsymbol{X}$ with mean and variance of σ^2.

In this way, Gibbs samplers for $\boldsymbol{A}$ and $\boldsymbol{S}$ were modelled assuming the initial value of noise variance is known. The sampling was started at the known matrices $\boldsymbol{A}$ and $\boldsymbol{S}$ as the test data. The sampler was run for N iterations. The product of the Sampled $\boldsymbol{A}$ and sampled $\boldsymbol{S}$ was close to $\boldsymbol{X}$ as it should be.

After $\boldsymbol{A}$ and $\boldsymbol{S}$ Noise Variances for each were modelled taking reference from equation 47 to 50 of the Moussaoui et al. (2006) paper. The posterior conditional of each noise variance (α_i^2) can be expressed as:

$$p(1/\alpha_i^2 | x_{(i,1:N)}, a_{(i,1:m)}^{(r+1)}, s_{(1:n,1:N)}^{(r+1)}) \propto (1/\alpha_{\mathrm{i}}^2)^{N/2} \, exp\{ \frac{-1}{2\alpha_{\mathrm{i}}^2} \sum_{\mathrm{t=1}}^{\mathrm{N}} (x_{(i,t)} - \sum_{\mathrm{k=1}}^{\mathrm{n}} a_{(i,k)}^{(r+1)} S_{(k,t)}^{(r+1)})^2 \} \times p(1/\alpha_{\mathrm{i}}^2) \tag{14}$$

The posterior density is given by equation (49) of the paper Moussaoui et al. (2006) and expressed as:

$$p\left(\frac{1}{\alpha_i^2}\middle|x_{(i,1:N)}, a_{(i,1:N)}^{(r+1)}, s_{(i,1:N)}^{(r+1)}\right) = \mathcal{G}(1/\alpha_i^2; \alpha_{\alpha_i^2}^{post}, \beta_{\alpha_i^2}^{post}) \tag{15}$$

Where,

$$\alpha_{\alpha_i^2}^{post} = \frac{N}{2} + \alpha_{\alpha_i^2}^{prior} \tag{16}$$

$$\beta_{\alpha_i^2}^{post} = \frac{1}{2}\sum_{t=1}^{N}(x_{(i,t)} - \sum_{k=1}^{n} a_{(i,k)}^{(r+1)}, s_{(k,t)}^{(r+1)})^2 + \beta_{\alpha_i^2}^{prior} \tag{17}$$

The choice of alpha prior and beta prior is made in such a way that all ranges of the noise variances are covered.

Case 2:

Sampling $\boldsymbol{A}, \boldsymbol{S}$ and $\boldsymbol{\sigma}$

Previously, source $\boldsymbol{S}$ and mixing coefficient $\boldsymbol{A}$ were sampled assuming noise level $\boldsymbol{\alpha}$ is known, so as to decrease the complexity of the problem and help during the verification of the algorithm. After the verification of the correctness of the samples with noise level set to initial value of 1. The code was later added to sample the noise variance. The code for sampling noise variance was added and it was used to sample the noise level $\boldsymbol{\sigma}$ along with source $\boldsymbol{S}$ and mixing coefficient $\boldsymbol{A}$. The variance function was derived using parameters alpha prior and beta prior which were derived from the gamma distribution so that both alpha prior and beta prior would be positive values.

4. RESULT AND DISCUSSION

After completing the method development process, the developed methods were tested on the test data. Firstly, the test data was generated using the developed function for the dimension of 2×2 matrix for two iterations on subtraction of the product of $\boldsymbol{A}$ and $\boldsymbol{S}$ from $\boldsymbol{X}$, the result obtained was small. This suggested that the results were consistent with the noise level.

After verification of correctness of method, the sampler was run for more iterations. The produced samples were not unique as suggested by the NMF. The convergence plot and histograms were plotted. The line and point plot of the samples for simple samples (i.e., 2×2 samples generated)

is shown below. The convergence plot generally shows how an error estimate during the solution process for nonlinear, and time step evolves in the time dependent, and parametric solvers.

A Test data set can be generated by sampling matrices $\boldsymbol{A}$ and $\boldsymbol{S}$ and adding noise to the resulting matrix $\boldsymbol{X}$. It makes m samples from n sources with N variables. This shows that the results are consistent with the noise level sigma.

$$\boldsymbol{X} = \begin{pmatrix} 4004904 & 1399595.9 \\ 1438149 & 413688.4 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 1801.7213 & 1039.2402 \\ 230.8338 & 828.7767 \end{pmatrix}$$

$$\boldsymbol{S} = \begin{pmatrix} 1455.789 & 582.4717 \\ 1329.795 & 336.9234 \end{pmatrix}$$

Test dataset generated using custom dataset generation function

Case 1: Assuming all hyperparameters are known

Gibbs Sampler for $\boldsymbol{A}$ and $\boldsymbol{S}$

Samples $\boldsymbol{A}$ and $\boldsymbol{S}$ under the assumption that all hyperparameters are known. The Gibbs sampler runs, assuming all hyperparameters are known for `N` iterations.

$$\boldsymbol{S} = \begin{pmatrix} 1455.789 & 582.4716 \\ 1329.795 & 336.9234 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 1801.7213 & 1039.2403 \\ 230.8338 & 828.7768 \end{pmatrix}$$

With an initial condition far from the true values Test Data $\boldsymbol{S}$ and $\boldsymbol{A}$ and run the Gibbs sampler for more iterations. We ran 1000 iterations and showed 100 iterations in the convergence plot (figure 2) for better visuals.

$$\boldsymbol{S} = \begin{pmatrix} 1316.699 & 206.3529 \\ 1116.125 & 671.1435 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 1722.9547 & 1555.642 \\ 770.5739 & 379.469 \end{pmatrix}$$

The result is not similar to Test Data $\boldsymbol{S}$ and $\boldsymbol{A}$ (remember that the solutions of NMF are usually not unique). But the resulting product $\boldsymbol{AS}$ is close to $\boldsymbol{X}$ as it should $\boldsymbol{X} - \boldsymbol{AS}$.

$$\boldsymbol{X} - \boldsymbol{AS} = \begin{pmatrix} -0.0003636898 & -0.15141291 \\ 0.1192367356 & -0.02005078 \end{pmatrix}$$

Considering the average of result,

$$\boldsymbol{X} - \boldsymbol{AS} = \begin{pmatrix} -0.0255696 & -0.02258240 \\ 0.1669948 & 0.06117259 \end{pmatrix}$$

The below convergence plot shows the first 100 iterations of sampled matrices $\boldsymbol{A}$ and $\boldsymbol{S}$.

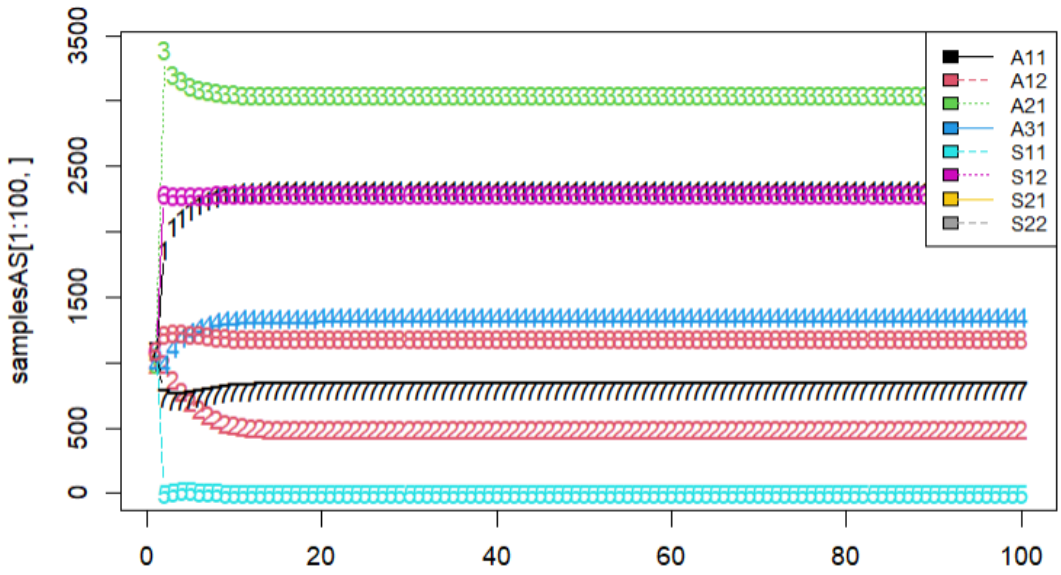

Figure 2: Convergence plot for the sampled matrices AS

The plot above shows that the Gibbs sampler quickly moves towards a particular solution for the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$. The convergence plot was generated to see the behaviour of the Gibbs sampler. The Gibbs sampler quickly moves towards a particular solution for the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$. Looking at the convergence plot in more detail, the sampler seems to move very slowly. This shows that the Gibbs sampler is very inefficient. For better visualization, the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$ of the first 50 iterations were taken and shown in the plots below. The histogram of the sample's various elements was plotted.

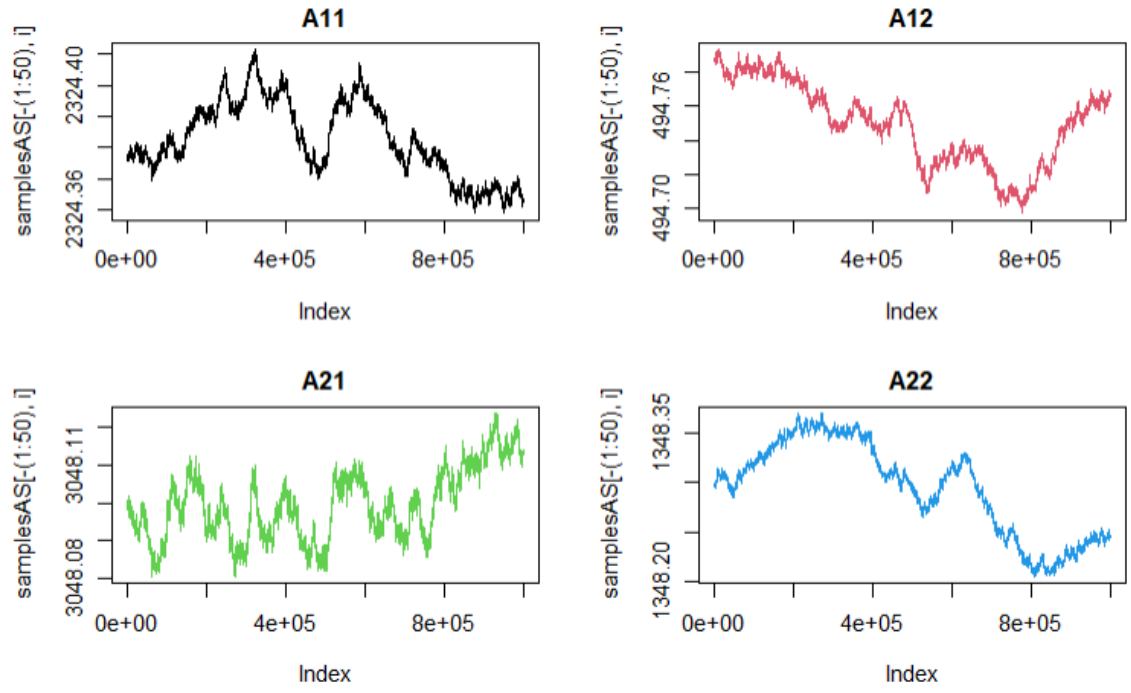

Figure 3: Convergence plot of the sample generated from Gibbs sampler for the sampled ***A***

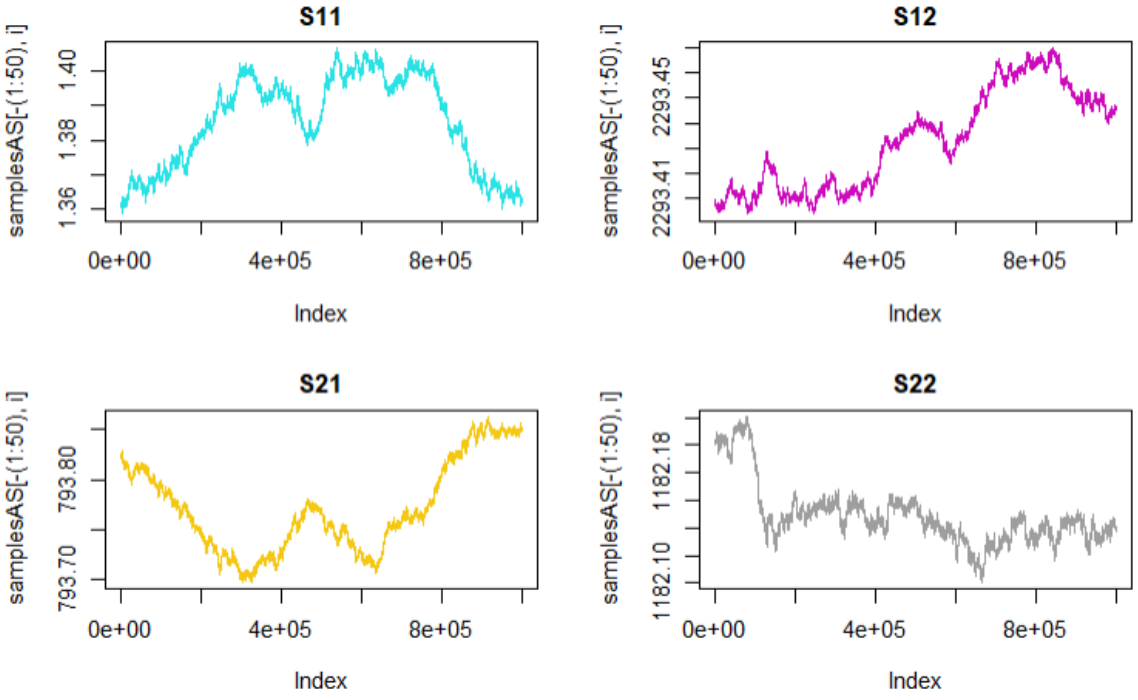

Figure 4: Convergence plot of the sample generated from Gibbs sampler for the sampled ***S***

Looking at the convergence plot in more detail, the sampler seems to move very slowly. This shows that the Gibbs sampler is very inefficient. From the analysis of the convergence plot

it seems like a random walk data. Some convergence plots are monotonously increasing e.g.: $A21$ and $S12$. While some are monotonously decreasing like: $S22$, $A21$. But other parameters are showing random patterns like sometimes the values are increasing and then decreasing.

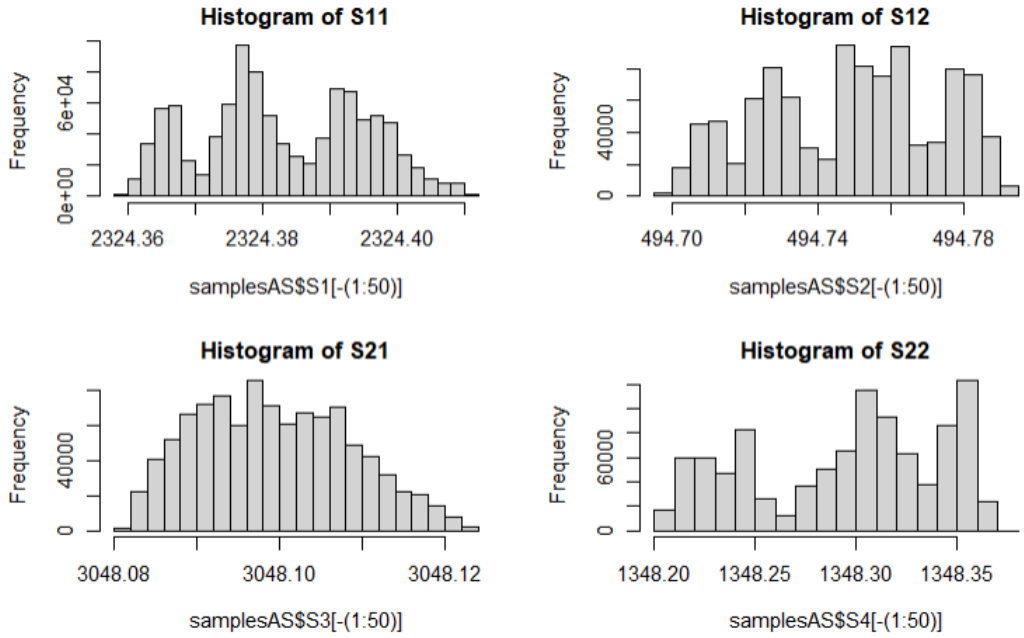

Figure 5: Histogram of S Matrix

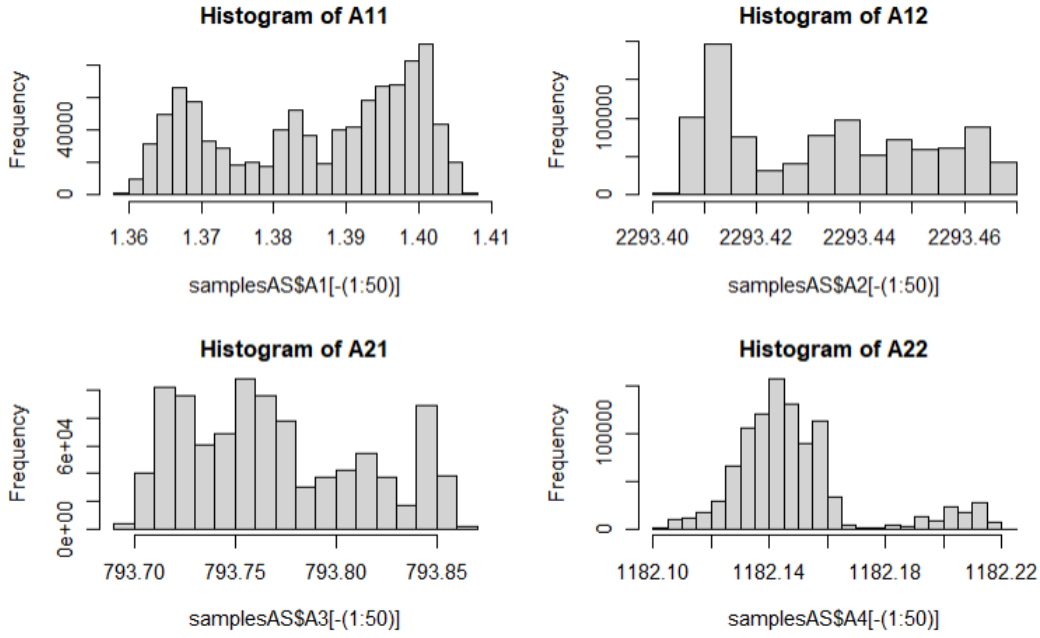

Figure 6: Histogram of A Matrix

The value of $\boldsymbol{A}$ and $\boldsymbol{S}$ sampled from Gibbs sampler is totally different from that of the NMF but the product of $\boldsymbol{A}$ and $\boldsymbol{S}$ in both cases are very close to that of test data. Infact the difference of the product of $\boldsymbol{A}$ and $\boldsymbol{S}$ obtained from NMF, and test data is 0 while the value is close to 0 for Gibbs sampler suggesting that Gibbs sampler is less precise or contains some noise. Despite the large number of samples, the shape of the histogram is very irregular. This is likely due to the fact that the Gibbs sampler samples quite inefficiently.

The algorithm was further tested on the simpler dataset such that the solution is unique. For this $\boldsymbol{A}$ and $\boldsymbol{S}$ were both diagonal (this is an old example, so only $\boldsymbol{S}$ is sampled, assuming $\boldsymbol{A}$ is already known). The solution was found by using the nmf function from the NMF package. The histogram of the certain elements of the sample was plotted which produced the following output.

Case 2: Sampling $\boldsymbol{S}$, $\boldsymbol{A}$ and $\boldsymbol{\sigma}$

The column means and standard deviations were also calculated. The means were close to the expected values and the standard deviations were also on the order of magnitude of what was expected from the noise added. Further, the results of the Gibbs sampler for sampling the source, mixing coefficient and sigma.

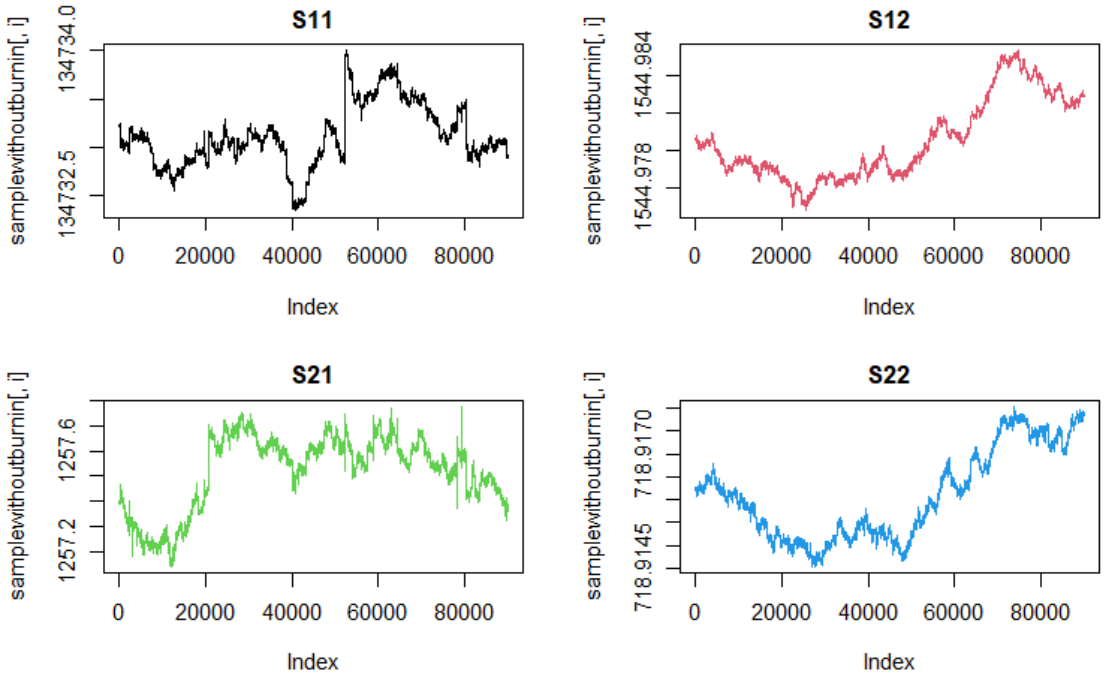

Figure 7: Convergence plot of the sampleASsigma for the sampled $\boldsymbol{S}$

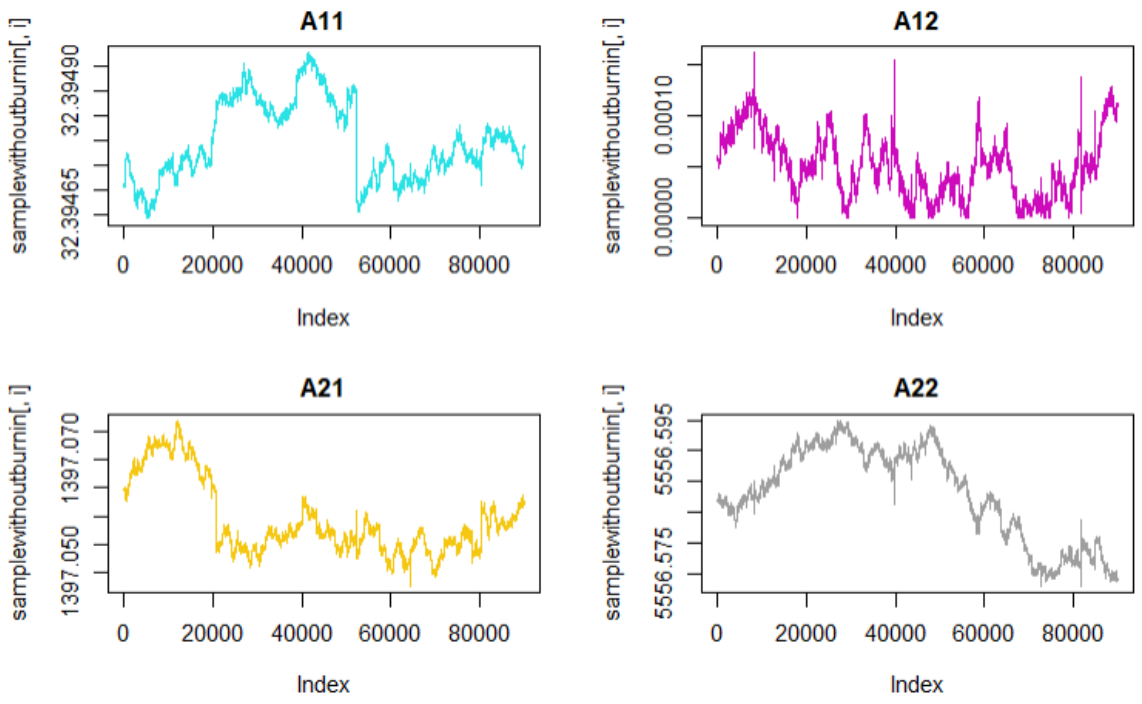

Figure 8: Convergence plot of the sampleASsigma for the sampled ***A***

The above convergence plots show that the sampler seems to move very slowly. This indicates that the Gibbs sampler is very inefficient. Some convergence plots are monotonously increasing from the analysis, e.g., $A12$ and $S22$. At the same time, some are monotonously decreasing, like $S21$ and $A22$. But other parameters are showing random patterns, like sometimes the values are increasing and then decreasing.

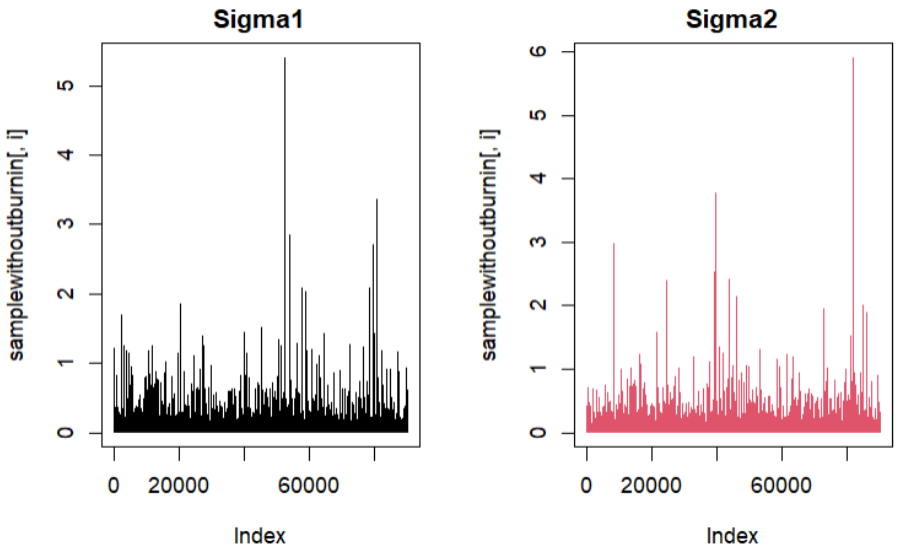

Figure 9: Convergence plot of the sampleASsigma for the sampled Sigma1 & Sigma2

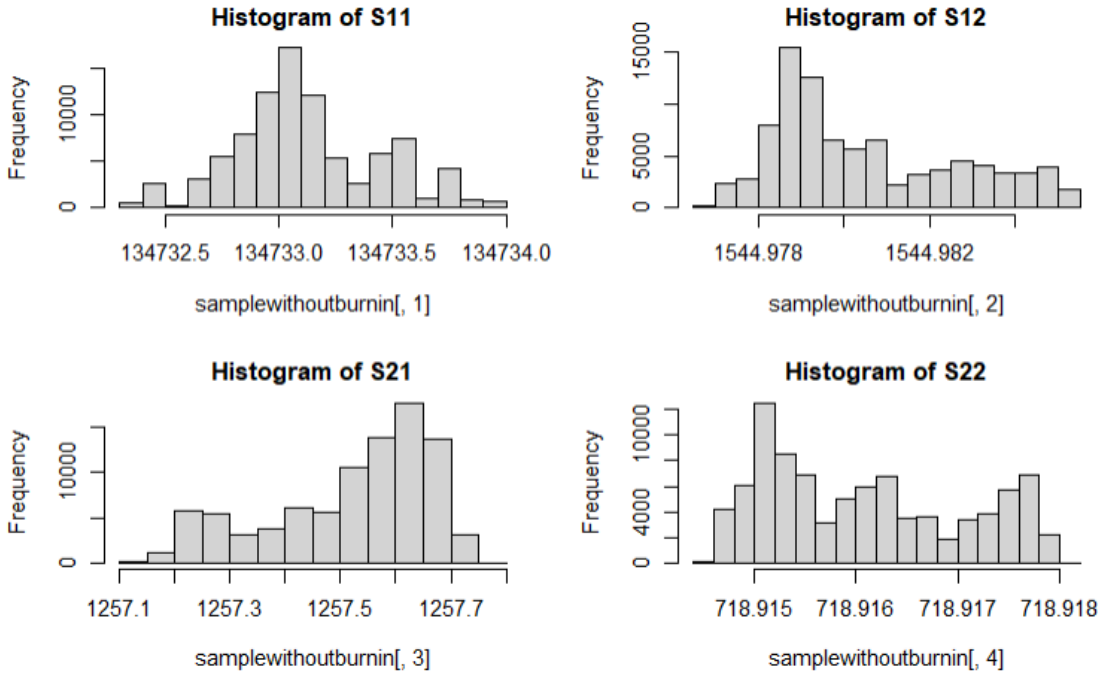

Figure 10: Histograms of S Matrix of SampleASsigma

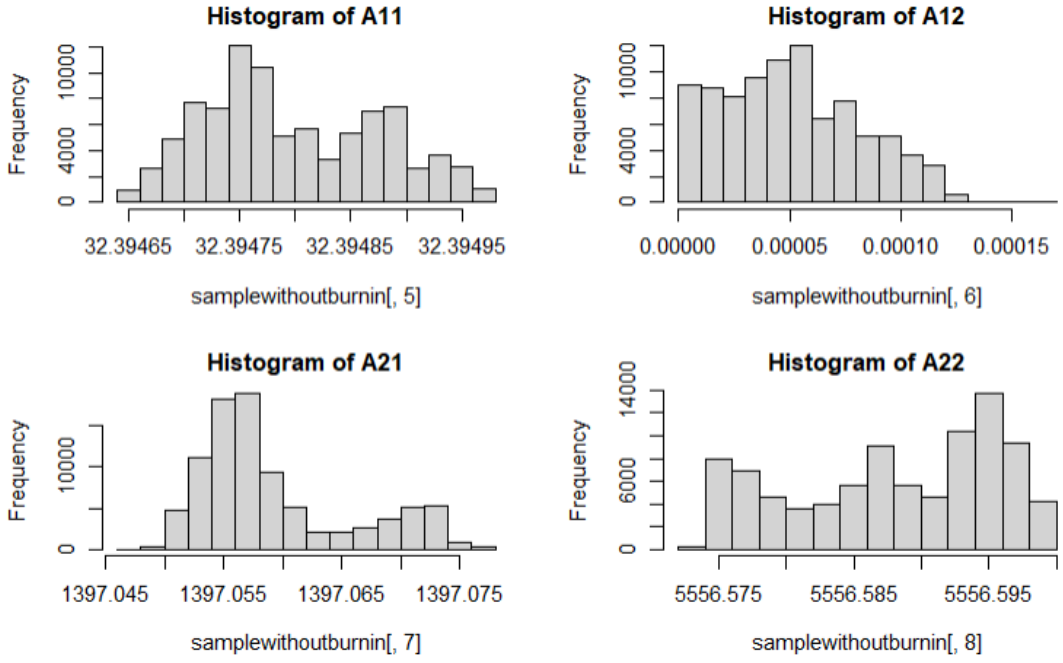

Figure 11: Histograms of A Matrix of SampleASsigma

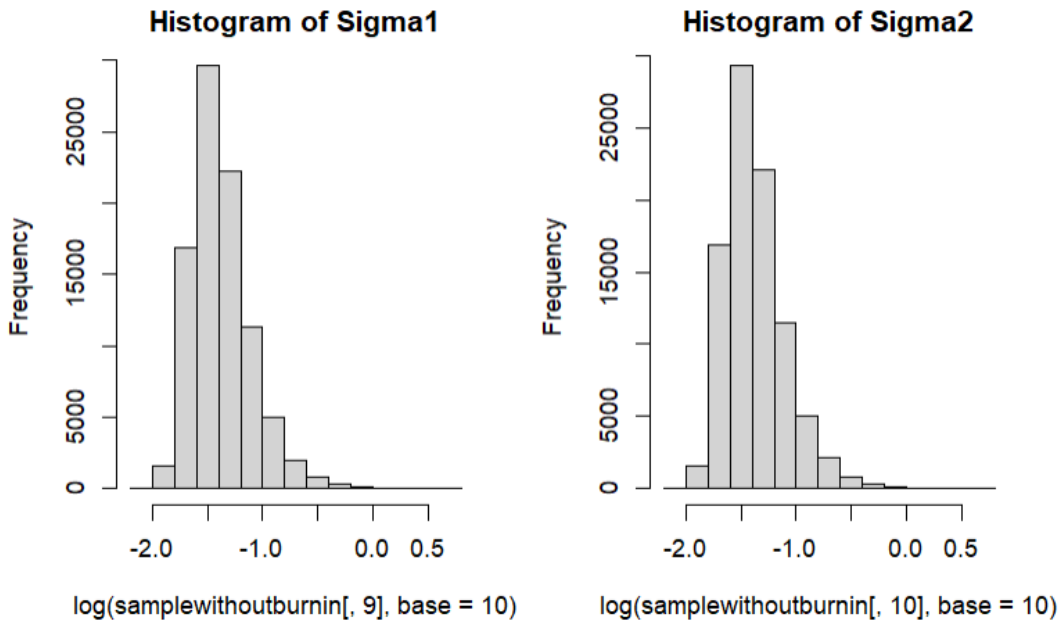

Figure 12: Histograms of Sigma1 and Sigma2

$$\boldsymbol{S} = \begin{pmatrix} 1455.794 & 582.4762 \\ 1329.791 & 336.9196 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 1801.7057 & 1039.2545 \\ 230.8399 & 828.7722 \end{pmatrix}$$

$$\boldsymbol{\sigma} = \begin{pmatrix} 0.02179184 & 0.07767759 \end{pmatrix}$$

Sampling S, A and sigma

The test produced certain expected as well as unexpected results. In the first phase the Gibbs sampler was developed solely based on the equation listed in the Moussaoui et al. (2006). Using these source signals $\boldsymbol{S}$ was sampled. But the sampled source was drastically different from the original source signal which was very unlikely and strange. On further inspection and debugging, it was found out that the equation mentioned in the paper had some mistakes which needed corrections. After the correction was made step by step, verification was done for each function developed.

Image Dataset Results

The algorithm was further tested on the Image dataset. The dataset containing 2000 images each with the dimension of 50×50 was formed by mixing four different images. Thus, the number of sources was four for the dataset. These sources were linearly mixed. The algorithm was run for ten iterations on the Image dataset. It takes longer than usual time. The first and tenth iteration output presented below. The difference was not much noticeable by looking into the images. But, on plotting the scatter plot of the pixels, the difference could be visualised, which can be seen below.

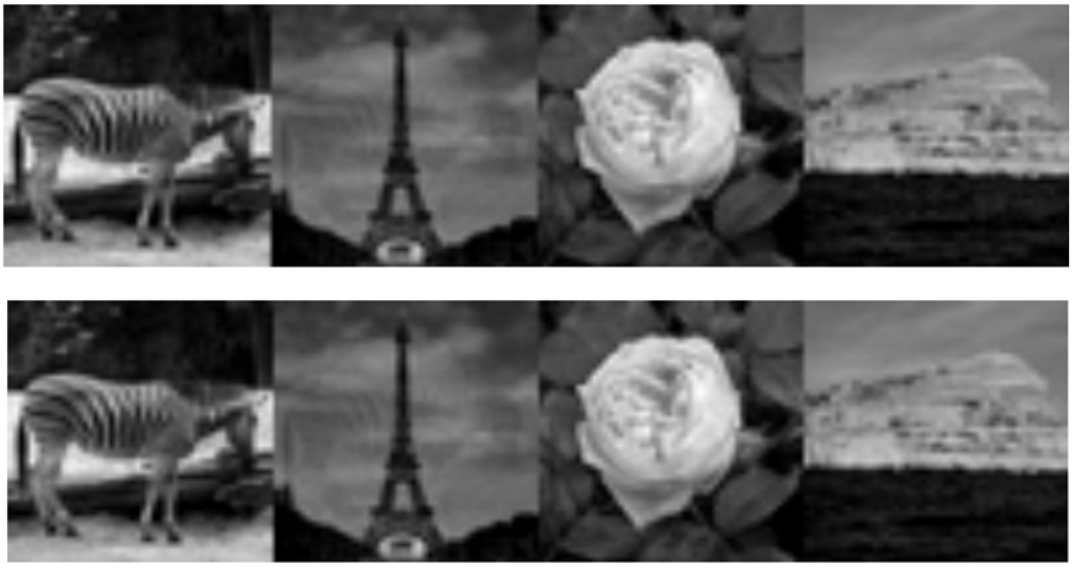

Figure 13: First & Tenth Iteration result from the Image Dataset

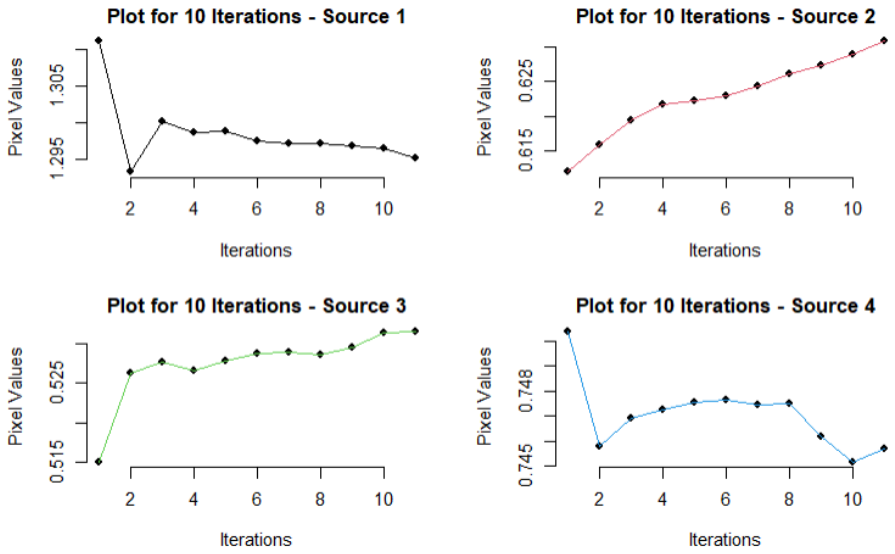

Figure 15: Results of plots for the ten iterations of four sources from Image Dataset

5. CONCLUSION AND FUTURE ENHANCEMENTS

The algorithm was successfully coded as per the equations mentioned in the paper. After completing the code, initially, the outputs were tested on Test Data. Many conclusions can be drawn from the experiment. The experiment produced expected as well as some unusual results. Analysing the paper further revealed that some equations written in the paper needed corrections.

These mistakes in the paper might have resulted during the paper's translation. After correction was made, the code was further tested, producing the expected output. The test dataset generated using the algorithm was consistent with the noise level. The histograms and line point plots produced also seemed correct. So, from this, it can be concluded that the algorithm mentioned in the paper by Moussaoui et al. (2006) has been implemented correctly, and it can be assured from the testing that the algorithm mentioned in the paper can be used for the sampling of non-negative sources. The dataset generated from the gamma distribution was consistent with the noise level $\boldsymbol{\sigma}$ signifying the code for generating the test data is correct. Besides, the image dataset was also formed by linear mixing of multiple non-negative sources. Thus, both datasets were proper for the sampling procedure used in the experiment. Gibbs sampling procedure was also implemented after reading various materials and literature papers and after gaining a proper knowledge algorithm. Due to this, the implementation of the algorithm was somewhat reliable and robust. The correctness of the algorithm is also later assured through testing with the test data. This was done differently for sampling $\boldsymbol{A}$ and $\boldsymbol{S}$. For $\boldsymbol{S}$. The true solution was used as the initial condition. The result produced by the sampler was very similar to the initial condition signifying the correctness of the sampler for $\boldsymbol{S}$. $\boldsymbol{A}$ new $\boldsymbol{S}$ was used, and the product of $\boldsymbol{S}$ and $\boldsymbol{A}$ was computed and compared with $\boldsymbol{X}$. The product remained close to $\boldsymbol{X}$ thereby assuring the sampler for $\boldsymbol{A}$ to be correct. All verifications were appropriately done for the algorithm.

There is always room for improvement of the project as well as the report. Lots of future enhancements can be done with this project.

Few enhancements that can be done are listed below:

- The algorithm can perform more iterations on the Image dataset, but due to time constraints, it was only evaluated for ten iterations.
- The algorithm can further test into a more complex dataset (maybe larger images with higher dimensions) due to the limited time the algorithm was tested on the test dataset that was simpler (i.e., 2000 images).

- Also, the computation time taken by the algorithm was pretty high. This might have been caused due to some redundant steps in the code. Thus, the quality of the code can be made better.
- Further, the method can be extended for NMF to convex-NMF.
- A better MCMC method for non-negative factorization can be developed, continuing from work done during this project.
- The findings of the work can be further applied to real-world problems like unsupervised learning.

6. SELF EVALUATION

The project has been very fruitful in terms of my career growth. This project was one of the most challenging projects I have done during my master's degree. The project was challenging in the sense that the algorithm development by taking the reference from this paper was very hard because of the fact that certain equations mentioned in the paper were not correct. The process of finding the issues with these equations was very tedious and time-consuming. But at last, the mistakes in the code were rectified with the help of my supervisor, and the code was implemented successfully. All steps are applied according to the instructions mentioned in the paper, and each equation after coding was subjected to unit testing to ensure the algorithm's correctness. From this project, I gained in-depth knowledge of the methods like NMF, Gibbs Sampling, and MCMC algorithms. I hope the knowledge gained through this project will help me in my future endeavors. Furthermore, I had a hard time testing the code with the image dataset as the testing process with the image was computationally expensive and took a lot of time. Thus, I gained technical knowledge from this project and learned the importance of project scheduling and project management. Besides, I have learned about various graphs for data visualization from the project, like convergence plots in more detail. This data visualization technique will help in my career as a data scientist. During the project period, I felt that the project would have been a lot easier if there weren't any errors in the equations mentioned in the paper.

REFERENCES:

Lopes, N., Ribeiro, B. (2015). Non-Negative Matrix Factorization (NMF). In: Machine Learning for Adaptive Many-Core Machines - A Practical Approach. Studies in Big Data, vol 7. https://doi.org/10.1007/978-3-319-06938-8_7

Colyer, A. (2019). The why and how of non negative matrix factorisation. The morning paper. https://blog.acolyer.org/2019/02/18/the-why-and-how-of-nonnegative-matrix-factorization/

Duarte, L. & Moussaoui, Said & Jutten, Christian. (2014). Source Separation in Chemical Analysis : Recent Achievements and Perspectives. Signal Processing Magazine, IEEE. 31. 135-146. 10.1109/MSP.2013.2296099

MOUSSAOUI, S., BRIE, D., MOHAMMAD-DJAFARI, A. & CARTERET, C. 2006. Separation of non-negative mixture of non-negative sources using a Bayesian approach and MCMC sampling. Ieee Transactions on Signal Processing, 54, 4133-4145.

Peharz,R & Pernkopf, F. (2012). Sparse non negative matrix factorisation with ℓ0-constraints. Neurocomputing. Vol 80.pp. 38-46. https://doi.org/10.1016/j.neucom.2011.09.024.

Ke-Lin, D & Swamy, M.N.S. (2019). Neural Networks and Statistical Learning. Second Edition. Xonlink Inc. Hangzhou, China. https://doi..org/10.1007/978-1-4471-7452-3.

Sengupta, S. (2020). Gamma Distribution Explained | What is Gamma Distribution?.Great Learning. Available at: https://www.mygreatlearning.com/blog/gamma-distribution/ . Visited on: 12-oct-2021

Schmidt, M.N., Winther, O. and Hansen, L. (2009). Bayesian non-negative matrix factorization.

Zavaleta, C.L., Smith, B.R., Walton, I., Doering, W., Davis, G., Shojaei, B., Natan, M.J. and Gambhir, S.S. (2009). Multiplexed imaging of surface enhanced Raman scattering nanotags in living mice using noninvasive Raman spectroscopy. Proceedings of the National Academy of Sciences, 106(32), pp.13511–13516.

Zhong, M. and Girolami, M. (2009). Reversible Jump MCMC for Non-Negative Matrix Factorization.

Robert, C.P. (1995). Simulation of truncated normal variables. Statistics and Computing, 5(2), pp.121–125.

Robert, C.P. (2007). The bayesian choice: From decision theoretic foundations to computational implementation. New York: Springer Science Bussines.

Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics, 5(2), pp.111–126.

Lee, D.D. and Seung, H.S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature, [online] 401(6755), pp.788–791. Available at: https://www.nature.com/articles/44565.

RStudio (Version 2021.09.0+351, R 3.0.1+). RStudio. [online] RStudio. Available at: https://www.rstudio.com/

S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, no. 6, pp. 721-741, Nov. 1984, doi: 10.1109 / TPAMI. 1984. 4767596.

Taboga, M. (2021).Markov Chain Monte Carlo (MCMC) methods. Lectures on probability theory and mathematical statistics. Kindle Direct Publishing. https://www.statlect.com/fundamentals-of-statistics/Markov-Chain-Monte-Carlo

Agrahari, S. 2021. Monte Carlo Markov Chain (MCMC), Explained.[onlin]. Towards Data Science. https://towardsdatascience.com/monte-carlo-markov-chain-mcmc-explained-94e3a6c8de11